Denoising your Monte Carlo Renders: Recent Advances in Image-Space Adaptive Sampling and Reconstruction

SIGGRAPH 2015 Course

Tentative Course Notes

Final course notes available at http://dx.doi.org/10.7919/F4H41PBV

Organizers

Pradeep Sen
University of California, Santa Barbara

Matthias Zwicker
University of Bern

Lecturers

Pradeep Sen
University of California, Santa Barbara

Matthias Zwicker
University of Bern

Fabrice Rousselle
Disney Research Zurich

Sung-Eui Yoon
Korea Advanced Institute of Science and Technology (KAIST)

Nima Khademi Kalantari
University of California, Santa Barbara

SIGGRAPH 2015 Courses, August 09 – 13, 2015, Los Angeles, CA.
ACM 978-1-4503-3634-5/15/08.

1 Abstract

Monte Carlo integration is firmly established as the basis for most practical realistic image synthesis algorithms because of its flexibility and generality. However, the visual quality of rendered images often suffers from estimator variance, which appears as visually distracting noise. The current shift in the computer graphics industry towards Monte Carlo rendering has sparked renewed interest in effective, practical noise reduction techniques that are applicable to a wide range of rendering effects, and easily integrated into existing production pipelines.

In this course, we survey recent advances in image-space adaptive sampling and reconstruction (filtering) algorithms for noise reduction, which have proven effective at reducing the computational cost of Monte Carlo techniques in practice. These techniques reduce variance by either controlling the sampling density over the image plane, and/or aggregating samples in a reconstruction step, possibly over large image regions in a way that preserves scene detail. To do this, they apply statistical techniques to sets of samples to drive the adaptive sampling and reconstruction process. In some cases, they use the statistical analysis to set the parameters for filtering. In others, they estimate the errors of several reconstruction filters, and select the best filter locally to minimize error. In this course, we aim to provide an overview for practitioners to assess these approaches, and for researchers to identify open research challenges and opportunities for future work.

In an introduction, we will first situate image-space adaptive sampling and reconstruction in the larger context of variance reduction for Monte Carlo rendering, and discuss its conceptual advantages and potential drawbacks. In the next part, we will provide details on five specific state-of-the-art algorithms. We will provide visual and quantitative comparisons, and discuss advantages and disadvantages in terms of image quality, computational requirements, and ease of implementation and integration with existing renderers. We will conclude the course by pointing out how some of these techniques are proving useful in real-world applications. Finally, we will discuss directions for potential further improvements.

This course brings together speakers that have made numerous contributions to image space adaptive rendering, which they presented at recent ACM SIGGRAPH, ACM SIGGRAPH Asia, and other conferences. The speakers bridge the gap between academia and industry, and they will be able to provide insights relevant to researchers, developers, and practioners alike.

Intended audience
Industry professionals and researchers interested in recent advances in image space adaptive sampling and reconstruction for reducing the noise of Monte Carlo rendering.

Prerequisites
Familiarity with rendering and with basic concepts of Monte Carlo integration as it is implemented in modern rendering systems is expected.

Level of difficulty
Intermediate/Advanced

2 About the Lecturers

Pradeep Sen
Department of Electrical and Computer Engineering
University of California, Santa Barbara (USCB)
Santa Barbara, CA 93106, USA

psen@ece.ucsb.edu
`http://www.ece.ucsb.edu/~psen/`

Pradeep Sen is an Associate Professor in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara. He attended Purdue University from 1992 - 1996, where he graduated with a B.S. in Computer and Electrical Engineering. He then attended Stanford University where he received his M.S. in Electrical Engineering in 1998 in the area of electron-beam lithography. In 2000, he joined the Stanford Graphics Lab where he did research on real-time rendering and computational photography. He received his Ph.D. in Electrical Engineering in June 2006, advised by Dr. Pat Hanrahan. His research interests include image synthesis and computational photography, and he is a co-author of over 30 technical publications, including eight SIGGRAPH/SIGGRAPH Asia publications. Dr. Sen has been awarded more than $2.2 million in research funding, including an NSF CAREER award in 2009.

Matthias Zwicker
Institute of Computer Science and Applied Mathematics
University of Bern
CH-3012 Bern, Switzerland

zwicker@iam.unibe.ch
`http://www.cgg.unibe.ch`

Matthias Zwicker has been a professor at the University of Bern and the head of the Computer Graphics Group at the Institute for Computer Science and Applied Mathematics since September 2008. He obtained a PhD from ETH in Zurich, Switzerland. From 2003 to 2006 he was a post-doctoral associate with the computer graphics group at the Massachusetts Institute of Technology, and then held a position as an Assistant Professor at the University of California in San Diego from 2006 to 2008. His research focus is on signal processing for high-quality rendering, point-based methods for rendering and modeling, and data-driven modeling and animation. He has served as a papers co-chair and conference chair of the IEEE/Eurographics Symposium on Point-Based Graphics, and as a papers co-chair for Eurographics 2010. He has been a member of program committees for various conferences including ACM SIGGRAPH and Eurographics, and he is an associate editor for Computer Graphics Forum, the journal of the Eurographics Association. He is an experienced speaker, having presented research papers and taught tutorials at major computer graphics conferences including ACM SIGGRAPH and Eurographics.

Sung-Eui Yoon
Department of Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Korea 305-701

sungeui@cs.kaist.ac.kr
`http://sglab.kaist.ac.kr/~sungeui/`

Sung-Eui Yoon is an associate professor at KAIST (Korea Advanced Institute of Science and Technology), where he is leading the Scalable Graphics Lab (SGLab). His research interests include scalable graphics/geometric algorithms, large-scale model rendering/visualization, cache-coherent algorithms, collision detection, and other geometric problems. He received his PhD degree from the Department of Computer Science, University of North Carolina at Chapel Hill at 2005. He served a paper chair for ACM Symp. on Interactive 3D Graphics and Games and published more than 50 papers on rendering and related fields. Some of his work received best paper awards (or nominations) and national-wide recognitions. He also gave numerous tutorials on some of his works at ACM SIGGRAPH, Eurographics, and IEEE Visualization. He is currently a IEEE senior member.

Fabrice Rousselle
Rendering Group
Disney Research Zurich
8006 Zurich, Switzerland

fabrice.rousselle@disneyresearch.com
`http://zurich.disneyresearch.com/~fabricer/`

Fabrice Rousselle is a Post-Doctoral Associate at Disney Research Zurich. He obtained a B.Ing. in computer engineering from the Ecole Polytechnique de Montréal in 2000, a M.S. in computer science from the University of Montreal in 2007, and a PhD in computer science from the University of Bern in 2014. Fabrice also worked as a software developer on video editing software at Matrox Video Group from 2000 to 2005, and as a research assistant in the colorimetry lab at the EPFL, where he worked on spectral reflection models from 2007 to 2009. Fabrice has been working on adaptive Monte Carlo rendering for the last four years, leveraging image-based techniques to improve rendering times. His work on adaptive rendering was published in ACM Transactions on Graphics and in the Computer Graphics Forum.

Nima Khademi Kalantari
Department of Electrical and Computer Engineering
University of California, Santa Barbara (UCSB)
Santa Barbara, CA 93106, USA

nima@umail.ucsb.edu
`http://www.ece.ucsb.edu/~nima/`

Nima Khademi Kalantari is a Ph.D. student at UCSB working under the supervision of Dr. Pradeep Sen in the MIRAGE Lab. His research interests focus on the application of signal processing to computer graphics and vision. He has published 10 journal papers in different areas including sampling, rendering, and imaging.

3 Syllabus

- **Introduction** (15 min, *Sen*)
  Objective: provide a motivation for the problem, give an idea of the effectiveness of image space adaptive rendering
  - Overview: a history of Monte Carlo rendering, overview of Monte Carlo integration, source of variance in MC rendered images
  - Previous approaches for variance reduction
  - Introduction to adaptive sampling and reconstruction; early approaches for adaptive sampling and filtering MC noise; explain renewed interest in the area
  - Situating image space methods in the broader context of variance reduction techniques

- **Algorithms: State-of-the-Art Case Studies** (65 min)
  Objective: provide details on existing state-of-the-art Monte Carlo denoising algorithms; discuss their pros and cons in terms of image quality, computational requirements, ease of implementation and integration with existing renderers.
  - Leveraging general image denoising algorithms for MC denoising, based on work by Kalantari and Sen [1] (7 min, *Kalantari*)
  - Filtering the noise from the random parameters in Monte Carlo, based on work by Sen and Darabi [2, 3] (10 min, *Sen*)
  - Cross-bilateral and NL-means filtering with SURE-based error estimation and feature prefiltering, based on work by Rousselle et al. [4, 5] (20 min, *Rousselle*)
  - Local weighted regression with explicit bias and variance estimation, based on work by Moon et al. [6] (20 min, *Yoon*)
  - Using machine learning to learn optimal filter parameters, based on work by Kalantari et al. [7] (8 min, *Kalantari*)

- **Conclusions** (10 min, *Zwicker*)
  Objective: highlight impact on industry, research opportunities and open challenges
  - Summary of state-of-the-art work in adaptive sampling and reconstruction
  - Adoption by industry
  - Future research directions

4 Recent Advances in Image Space Adaptive Sampling and Reconstruction

This course builds on a state-of-the-art (STAR) report on adaptive sampling and reconstruction that was presented at Eurographics 2015 and will appear in the *Computer Graphics Forum*. The STAR report discusses both "a priori" and "a posteriori" methods for adaptive rendering. "A priori" methods analyze the light transport equations and derive sampling rates and reconstruction filters from this analysis, for example using frequency analysis or by estimating derivatives of light transport. In contrast, "a posteriori" methods apply statistical techniques to sets of samples, which may be acquired using a conventional renderer, to drive the adaptive sampling and reconstruction process.

This SIGGRAPH course focuses exclusively on "a posteriori' methods in image space that are tailored for high-end rendering. We believe these techniques are currently of most interest in the industry, and the restriction to a narrower area allows us to present a highly focused course on a timely topic. We provide an adapted version of the STAR report, focusing on image space techniques, as notes for the proposed course.

At the end of this document, we have also appended our tentative slides for the course. The final slides and all course materials will be available on the course webpage at:
`http://dx.doi.org/10.7919/F4H41PBV`

4.1 Introduction

Today, Monte Carlo methods are widely accepted as the most practical methods for realistic image synthesis. The rendering equation [8] formulates this problem as an integral over all light paths that connect any point on a light source to a point on an image sensor. Monte Carlo methods estimate this integral by randomly sampling light paths and accumulating their image contributions. Even simple Monte Carlo rendering algorithms come with a number of very desirable properties: they are consistent, which means that as the number of sampled paths increases, the estimated image converges to the correct solution; some algorithms, like (bidirectional) path tracing, are also unbiased, that is, the expected value of the estimated image corresponds to the correct solution and the error consists only of variance; and finally, they are applicable to most scene configurations that are relevant in practice.

On the other hand, because only a limited number of random light paths can be sampled to compute each image, all Monte Carlo methods suffer from variance in the estimated pixel values, which appears as image noise. Unfortunately, computation times to obtain visually satisfactory results without noticeable noise are often in the minutes and hours. Therefore, researchers have proposed a wide variety of noise or variance reduction strategies over the years, from different path sampling strategies (importance sampling, bidirectional techniques, Metropolis sampling) to statistical techniques (quasi-Monte Carlo sampling using low-discrepancy sequences, density estimation, control variates), or signal processing methods (frequency analysis, non-linear filtering), to name the most prominent ones. In this paper, we survey recent advances in adaptive sampling and reconstruction, which have proven very effective at reducing the computational cost of Monte Carlo techniques in practice.

The amount of variance generally varies greatly between local regions of rendered images. Adaptive sampling refers to techniques that control sampling densities based on previously acquired samples, in contrast to sampling predetermined target densities, to distribute samples according to the local amount of variance. Adaptive reconstruction computes output pixel values using locally defined reconstruction filters. These filters reduce variance by sharing information between pixels, and by aggregating samples over larger image regions, while trying to avoid blurriness. Adaptive sampling generally requires an adaptive reconstruction

step, while adaptive reconstruction may also be performed without adaptive sampling. A common strategy is to sample and reconstruct iteratively, where the estimated reconstruction error determines sampling densities in the next step.

Pioneering efforts in adaptive sampling and reconstruction occurred almost simultaneously with the development of the first Monte Carlo algorithms. For example, Mitchell [9] proposed a two-step approach to adaptively sample the image plane considering a contrast metric inspired by human perception, and he developed a reconstruction filter to deal with the nonuniform sample distributions. Parker and Sloan [10] and Guo [11] sample the image plane using progressive refinement, and both apply polynomial reconstruction filters. Ward et al.'s irradiance caching algorithm [12] sparsely and adaptively samples irradiance in the image plane in a greedy fashion, and uses a custom tailored reconstruction strategy to interpolate irradiance at each pixel. Inspired by these works, other researchers have considered more advanced perceptual error estimates [13, 14], or alternative reconstruction strategies such as splatting [15] and anisotropic diffusion [16]. Despite these early successes, recent advances in adaptive sampling and reconstruction have been significant, reducing the number of samples often by orders of magnitudes without sacrificing quality compared to these earlier methods.

As a starting point that sparked these advances, we would like to single out the work by Overbeck et al. [17], who estimate the error of a 2D wavelet approximation of the rendered image after distributing an initial set of samples, and then they iteratively add more samples. Here, sample densities and reconstruction filters are derived a posteriori from the statistics of a set of samples. Hence, we call such methods "a posteriori". While this strategy goes back to the earliest adaptive sampling techniques [9], Overbeck's approach combines it with more powerful reconstruction filters, which was a crucial step paving the way for further advances.

4.2 Image Space Adaptive Sampling and Reconstruction

Instead of analyzing the light transport equations to derive local signal properties analytically, *a posteriori* techniques statistically analyze sets of samples acquired using a Monte Carlo renderer with little additional information. At the minimum these methods require local, usually per-pixel, estimates of variance. We illustrate the generic template followed by many methods in this section in Figure 1. A key idea in most techniques is to use a family of reconstruction filters and develop error estimates for the filter outputs, usually using a mean squared error metric (MSE). In general, MSE can be expressed as a sum of squared bias and variance, where bias corresponds to blurriness, and variance is residual noise from the input Monte Carlo samples. A family of filters should provide a trade-off between bias and variance, and the goal is to locally select the best filter that optimally balances bias and variance to minimize MSE. In addition, the local error estimates also make it possible to control sampling densities. We organize our survey according to the types of filters that are used, distinguishing multiscale filters (Section 4.2.1), filters designed for generic image denoising (Section 4.2.2), and filters derived from auxiliary features (Section 4.2.3). Most approaches rely on image space filtering because of its simplicity and efficiency.

A notable exception is the work by Hachisuka et al. [19], where they store samples in multidimensional path space, which may include effects such as motion blur, depth of field, and soft shadows. They estimate the local integration error incurred by an initial set of samples, and adaptively distribute more samples where the error is highest in the multidimensional space. Hachisuka et al.'s method is agnostic of the underlying rendering effects, and as a consequence, it cannot match the quality of specialized techniques. On the other hand, this should in principle make it easier to apply the approach to arbitrary combinations of effects, but developing efficient algorithms with higher dimensional path spaces is challenging.

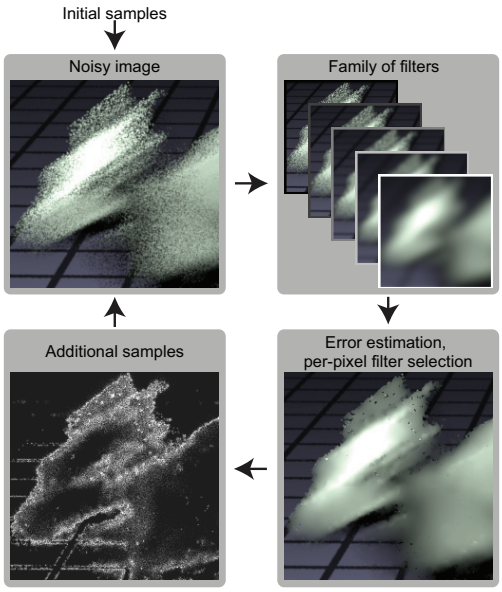


Figure 1: The generic template followed by many techniques discussed in Section 4.2. Typically, the iteration terminates when a given sample budget is exhausted. This figure is adapted from Rousselle et al. [18].

4.2.1 Multiscale Filters

A family of filters obtained by scaling the support of a basis filter provides a natural trade-off between bias and variance. Overbeck et al. [17] proposed an algorithm following the strategy outlined in Figure 1 using Daubechies wavelets. They perform wavelet analysis of the rendered image and analyze its error using Mitchell's contrast metric [9]. They perform wavelet shrinkage [20], that is, they subtract a noise estimate from individual wavelet coefficients, to minimize the error metric. In addition, they use the error metric to adaptively distribute samples in the image plane focusing on regions with high error. Rousselle et al. [18] observed that critically sampled wavelets lead to ringing artifacts when used for denoising. They obtain higher filtering quality using Gaussian filters at several scales. They perform an MSE analysis by estimating bias and variance separately, and they locally select the scale that minimizes the error. We visualize their approach in Figure 2.

The main advantage of such multiscale filters is that they are very efficient to compute, facilitating potential real-time applications. For example, Dammertz et al. [21] use wavelets for real-time filtering. On the other hand, families of filters parameterized by a single scale parameter require small scales to filter complex image structures without introducing blurriness. While including parameters to describe anisotropy should be possible, this may not be enough to reach the quality of methods that enable even more complex filter shapes, as described in the following sections.

4.2.2 Leveraging Image Denoising Filters

Image denoising filters developed by the image processing community to restore images corrupted by spatially uniform noise have recently achieved impressive results [22–24]. Hence, it seems promising to leverage these methods for adaptive sampling and reconstruction in Monte Carlo rendering.

Patch-based techniques compute denoising filters by assessing the similarity of local image patches. They

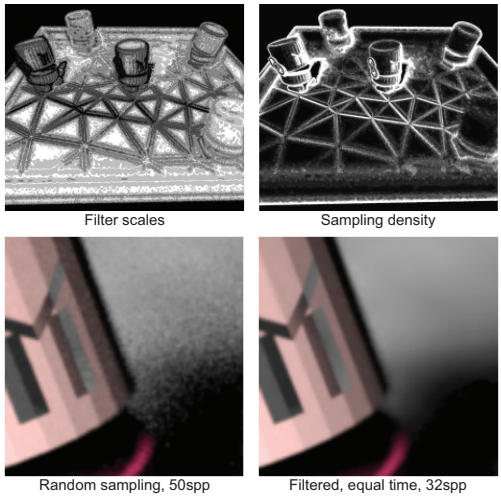


Figure 2: Visualization of Gaussian filter scales (top left), brighter levels indicating larger filters. Adaptive sampling (top right, brighter means more samples) places more samples where filters are small or where there is more noise due to defocus or soft shadow effects. The bottom row shows an equal time quality comparison to naive random sampling.

are currently among the most successful filters for generic image denoising. A key idea is that similar patches may be used for denoising within large neighborhoods, even globally over the whole image, without introducing blurriness. This justifies the term "non-local". The non-local means (NL-means) filter [23] embodies the simplest implementation of the strategy. It generalizes bilateral filtering [25], which Xu and Pattanaik [26] used for denoising in Monte Carlo rendering. Instead of weighting a neighboring pixel based on the difference of its value to the center pixel, NL-means determines the weight based on the similarity of small patches around the neighbor pixel and the center. Compared to bilateral filtering, NL-means is a much more effective denoising filter, since the filter weights are more robust to noise in the input. Rousselle et al. [4] exploit NL-means filtering for adaptive rendering, and modify it to cope with non-uniform noise levels. Unfortunately, a full error analysis of NL-means is challenging because the filter itself depends on the noisy input. Instead, Rousselle et al. only consider variance and neglect bias. They estimate output variance simply using the per-pixel difference of two independent filtered images. The per-pixel variance estimate then drives adaptive sampling.

Delbracio et al. [27] propose a similar non-local denoising approach. Instead of comparing patch similarity based on pixel values, however, they gather histograms of sampled values at each pixel, and assess patch similarity based on the differences of these histograms measured by the χ^2 distance. In addition, they propose a multiscale approach to remove low frequency noise. They demonstrate that their approach improves upon NL-means filtering.

Kalantari and Sen [1] propose a generic method (Adaptive Multilevel Denoising, "AMLD" in the following, see Table 1 for a list of acronyms) that is designed to operate with any denoising method for globally uniform noise, such as BM3D [22], assuming that it has a parameter to adjust the filter to the global noise level. First, they propose a novel per-pixel variance estimate using the median absolute deviation [20]. Then, to deal with spatially nonuniform variance in Monte Carlo rendering, they propose to denoise the complete rendered images several times, each time setting a different noise level. Finally, they compose the result image by selecting pixels from the filter outputs with appropriate noise levels.

The techniques discussed here significantly improve the output quality compared to simple multiscale filters. We show an example result in Figure 3. The methods mentioned here and in the previous subsection all

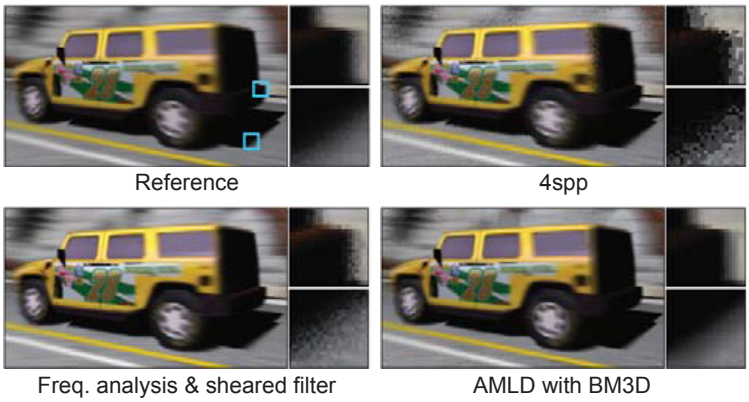


Figure 3: A scene with motion blur and soft shadows (top left: reference, to right: 4spp). The figure compares the sheared filter for motion blur derived from a frequency analysis [28] (bottom left) to adaptive reconstruction using generic image denoising filters [1] (AMLD, bottom right). Both use four samples per pixel (top right) as input.

come with the advantage that they require minimal changes to an existing renderer. Besides noisy renderings they only need per pixel statistics such as contrast [17] or variance [1, 4] estimates, or pixel histograms [27] as inputs.

4.2.3 Filters using Auxiliary Features

Besides noisy images, it is easy to save auxiliary image features such as per-pixel normals, depth, or diffuse reflectance from most Monte Carlo rendering systems. The feature images contain rich information about image structures that can be exploited to design adaptive filters that preserve these structures. An early approach by McCool [16] used depth and normal information to control anisotropic diffusion. More recently, researchers explored the cross-bilateral filter [29, 30]. Here, the core idea is to derive the weight with which a neighbor contributes to the filter output at a center pixel from the pairwise feature differences.

Sen and Darabi [2, 3] observed that the relation between the feature differences and filter weights should depend on the degree to which the feature determines the true pixel value. After all, certain features are highly dependent on the random parameters of the Monte Carlo process and are therefore not reliable for filtering. They propose to measure this dependency by computing the mutual information between sample features and the random parameters used to compute them. Their approach (Random Parameter Filtering, or "RPF" in the following, see also Table 1) naturally handles noisy features, which occur when rendering effects such as motion blur or depth of field.

A disadvantage of this approach is that the complexity of the algorithm depends on the number of samples and can become expensive for more than a few dozen samples. In addition, while RPF is based on useful intuition about the influence that features should have on the filter, it does not try to minimize the output error directly, and methods based on explicit error estimation have proven superior at high sampling rates. However, at low sampling rates (e.g., less than 16spp), the ability of RPF to detect noisy features can yield improvements over other methods (see Figure 4). Furthermore, Park et al. [32] subsequently proposed a modification of RPF whose complexity is independent of the number of samples without significant quality degradation.

Li et al. [31] first introduced Stein's unbiased risk estimator (SURE) to adaptive sampling and reconstruction in rendering. SURE is a general technique to estimate the accuracy of a statistical estimator. Van De Ville

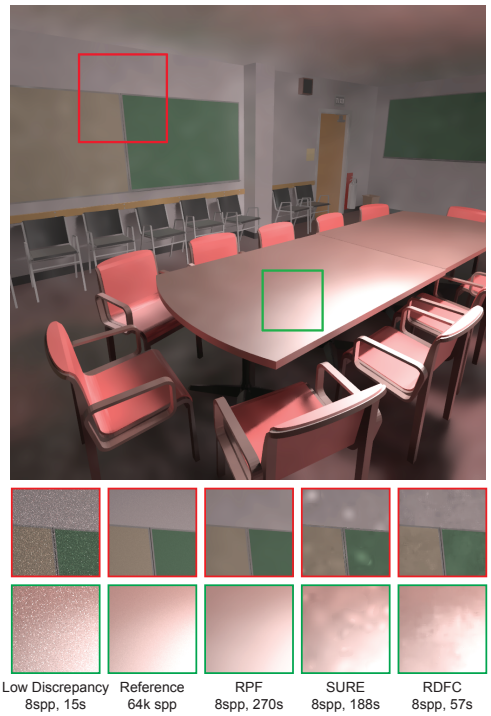


Figure 4: Comparison of RPF [3] (main image) with SURE [31] and RDFC [5] (in close-ups) at a low, equal sampling rate (8spp). RPF is most effective at such low sampling rates.

and Kocher [33] applied it to estimate global parameters in NL-means image denoising, and Li et al. showed how to leverage it to obtain local, unbiased MSE estimates of cross-bilateral filters. They use a set of cross-bilateral filters with spatial support windows of varying sizes, and select the best size at each pixel by picking the filter leading to the smallest estimate of MSE using SURE. Their approach ("SURE" in the following, see also Table 1) typically outperforms RPF presumably because it selects filters according to an actual error estimate. In Figure 5 we show an equal-sample comparison between SURE, the axis-aligned filtering technique based on *a priori* frequency analysis by Mehta et al. [34], and the approach by Kalantari and Sen (AMLD) using BM3D. While Mehta et al. use GPU raytracing, the other methods are implemented on top of PBRT [35]. The filtering overhead of SURE and AMLD is on the order of 30-40 seconds. The methods perform similarly at a fixed sampling rate, where SURE and AMLD tend to overblur, and Mehta has some residual noise.

Rousselle et al. [5] also build on SURE error estimation and cross-bilateral filtering. Instead of minimizing error by varying the spatial filter support, they carefully design three filters with the same support, but differing in their other parameters. One filter is tuned to be most sensitive to the noisy color information and disregards the features, one disregards color and is determined purely by the features, and the third is in between. These three filters lead to fewer outliers in the error estimate, since error estimation of small filters, such as in Li et al. [31], is very sensitive to noise. In addition, they propose a feature prefiltering step to deal with noisy features and introduce two new features to better represent soft shadows due to direct illumination and caustics. Figure 6 shows the improvements of their approach (Robust Denoising using Feature and Color Information, "RDFC" in the following, see also Table 1) over Li et al. (SURE).

Similar to the bilateral filter, the guided filter [36] can be applied for edge preserving image denoising. Given a denoising window, the ground truth image is locally approximated as a linear function of a guide image.

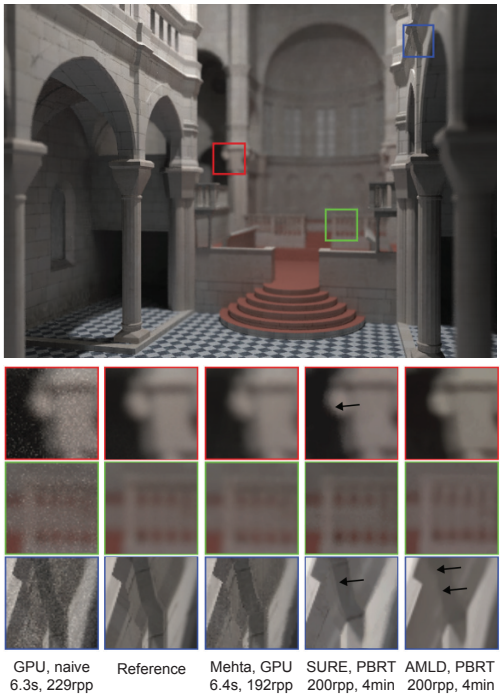


Figure 5: An equal-sample comparison in terms of ray segments per pixel (rpp) of Mehta et al.'s approach [34] based on *a priori* frequency analysis for soft shadows, indirect illumination, and depth of field (main image), with *a posteriori* techniques (in close-ups) by Li et al. [31] (SURE) and Kalantari and Sen [1] (AMLD).

The two coefficients of the linear function are estimated by minimizing the L_2 error between the linearly transformed guide and the noisy input image. Bauszat et al. [37] applied the guided filter to reduce noise generated by Monte Carlo rendering while preserving edges by using geometric features as guide images. Recently, Moon et al. [6] also used a linear model to approximate the ground truth function locally, but in addition they weigh the error of each pixel individually based on the features ("LWR" for Local Weighted Regression, see also Table 1). They introduce an MSE estimate by analyzing bias and variance separately, and they optimize the parameters used to determine the weights to minimize the MSE at each pixel. Finally, they deal with noisy features by performing dimensionality reduction via a truncated SVD (Singular Value Decomposition), and their optimization is solved on the local features with a reduced dimensionality. As shown in Figure 7, their approach achieves high-quality rendering results even in nonlinear functions such as glossy highlights and defocused areas thanks to the automatic parameter selection and the truncated SVD, respectively.

Most recently, Kalantari et al. [7] observed there is a complex relationship between the noisy scene data and the ideal parameters of filters such as cross-bilateral filters and proposed to learn this relationship using a nonlinear regression model. To do this, they used a multilayer perceptron neural network and combined it with a matching filter during both training and testing. They trained it in an offline process on a set of noisy images of scenes with a variety of distributed effects. Then at run-time, they used the trained network to drive the filter parameters for new test scenes to produce filtered images that approximate the ground truth. This learning-based filtering approach (LBF) produces high-quality results in only a few seconds that are superior to the previous methods, as shown in Fig. 8.

The methods discussed so far in this section use features that are obtained directly as byproducts of usual Monte Carlo rendering. As an interesting alternative, Moon et al. [38] proposed a new feature, a virtual

Figure 6: Equal-time comparison of Rousselle et al. [5] (RDFC, main image) and Li et al. [31] (SURE, in close-ups). RDFC is more robust to noisy error estimates, and it includes a feature to capture soft shadows from direct illumination, which leads to significant quality improvements.

flash image, which serves as an edge-stopping function. The virtual flash image is motivated by flash photography [29, 30] where a flash image provides high-frequency information of the noisy input image taken without a flash. To emulate the flash image in rendering, they put an additional light source on the viewing position and construct the virtual image through an actual shading. This simple idea provides a means to capture a wide set of edges such as reflected and refracted edges with little computational overhead since it is performed by reusing a subset of light paths. Furthermore, they define homogeneous pixels as neighboring pixels whose sample means are within a confidence interval of the true mean at the center pixel. By restricting the denoising to homogeneous pixels, they preserve additional edges (e.g., caustics) that the flash image does not include, and they obtain consistent denoising results that converge to the ground truth.

Table 1 summarizes the main characteristics of the *a posteriori* methods cited in this section. This table does not list the light transport effects supported, since all cited methods are generic (though some, like Dammertz et al. [21], give bad results if their assumption of noise free features is violated). A commonality of all methods discussed here is that they reconstruct pixel values by building (locally adaptive) weighted averages of complete Monte Carlo path samples. While this is particularly simple, it could be further generalized to aggregating partial paths that do not immediately connect a light source and the image sensor. For example, photon mapping [39] and its generalizations [40–42] aggregate several "light subpaths" (partial paths connected to light sources) when connecting to individual "eye subpaths" (partial paths connected to the image sensor). The lightcuts algorithm builds adaptive hierarchies of light subpaths, and connects these to individual [43] or groups of eye subpaths [44] while respecting a desired error bound. It may be interesting to investigate extensions of the algorithms summarized here to operate on light subpaths also, instead of requiring complete path samples.

There is also room to explore between the extremes of working in 2D image space, like most methods discussed here, and the full high-dimensional path space [19]. For example, deCoro et al. [45] represent samples in a joint image and color space to detect and reject outlier samples in this 5D space. Finally, while all reconstruction methods discussed here are driven (implicitly) by (weighted) L_2 errors, one could generalize this to formulations inspired by total variation methods or compressive sensing. They rely on sparse error norms that have proven to provide visually more convincing results, for example, in image restoration [46]. While some methods based on compressive sensing have been proposed for rendering [47, 48], it is likely

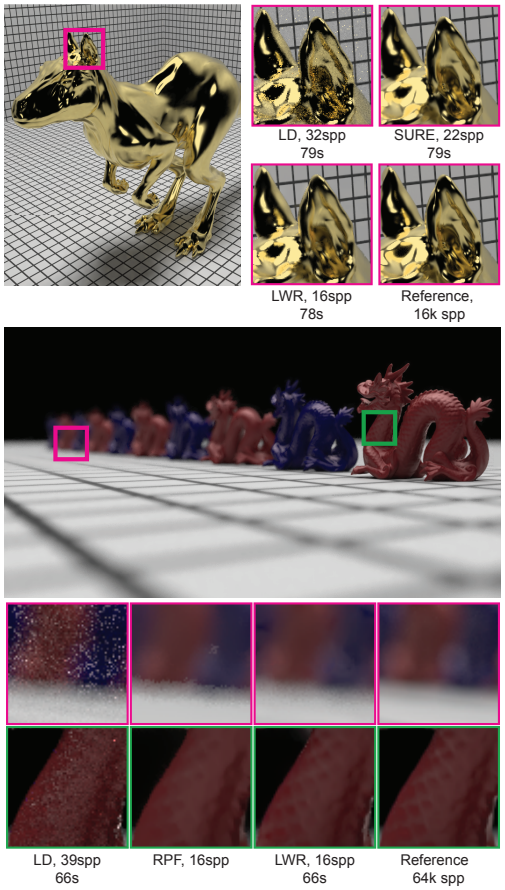


Figure 7: Moon et al. [6] (LWR, main images) use weighted local regression to estimate the denoised output. At equal render times, LWR shows better results on a variety of rendering effects compared to naive low-discrepancy sampling (LD), Li et al. [31] (SURE), and Sen and Darabi [3] (RPF, all in close-ups). This figure is modified from Moon et al. [6].

that the full potential of these techniques has not been realized yet.

4.3 Discussion and Conclusions

Image space "a posteriori" methods forego analytic analysis for statistical error estimation based on acquired samples. They are typically ignorant of the underlying rendering effects and applicable to arbitrary combinations of light transport phenomena. A key property of these methods is also that they rely on heuristically defined filters, such as cross-bilateral filters using auxiliary features, which can have arbitrary support shapes and extend over large image areas. Hence, these methods can be effective at removing noise even in image regions with intricate structures. In addition, many methods are consistent because their filters are eventually restricted by the empirical variance of rendered pixel means, which vanishes with increasing sample counts. It is also relatively straightforward in practice to modify existing renderers to support *a posteriori* adaptive sampling and reconstruction, since the required information such as feature images or pixel variances are available as a byproduct of conventional rendering. As a disadvantage, the most sophisticated reconstruction filters, like local weighted regression or methods requiring several passes of cross-bilateral filtering, are still too expensive for real-time applications even using GPU implementations.

The techniques discussed here may well have an impact on CG movie production. Until recently, rasterization-

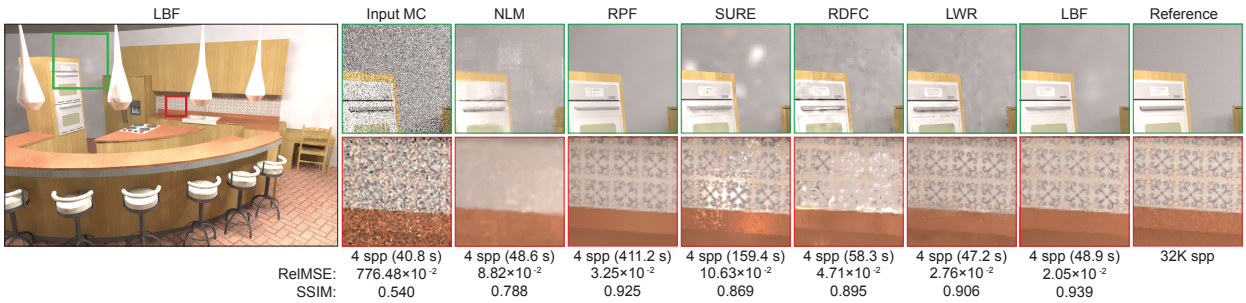


Figure 8: Comparison between learning-based filtering (LBF) [7] and several state-of-the-art algorithms on the KITCHEN scene rendered at 4 spp. Note that the ground truth image is still noisy even at 32K spp. NLM [4] is a color-based method which does not keep geometry or texture detail. RPF [3], SURE [31], RDFC [5], and LWR [6] use additional scene features such as world positions and shading normals to keep the details. However, they often do not weight the importance of each feature optimally, resulting in under/over blurred regions or splotches in the final result. LBF [7] preserves scene detail and generates a higher-quality, noise-free result faster than most other methods. The relative mean squared error (RelMSE) and structural similarity (SSIM) index are listed below each image. Larger SSIM values indicate better perceptual quality.

based pipelines, such as Pixar's REYES architecture, have been the norm, with the notable exception of Blue Sky Studios, which used ray tracing on all of its productions. However, we are currently seeing an industry-wide shift to physically based Monte Carlo rendering, spearheaded by Solid Angle's Arnold renderer that is now used in a wide array of productions. Pixar is introducing a new Monte Carlo rendering pipeline, RIS, in its RenderMan software. Walt Disney Animation Studios (WDAS) developed its own Monte Carlo renderer, Hyperion, and Weta Digital recently introduced its renderer, Manuka. These renderers come in addition to existing ones, such as Mental Ray from Mental Images (now NVIDIA) and V-Ray from Chaos Group, both of which are readily integrated in professional production tools. With this shift to Monte Carlo rendering comes a strong interest for robust adaptive rendering solutions, and denoising in particular. For example, Anderson and Meyer from Pixar proposed a statistical technique based on PCA to denoise animation sequences [50]. Goddard, from Image Engine, recently presented the denoising technique they developed for the production of Elysium [51]. WDAS also used denoising in the production of Big Hero 6, which proved to be crucial for reaching production deadlines, and Pixar is currently investigating various denoising alternatives for its own productions. These production environments bring new constraints and demands that stretch the capabilities of existing methods and will hopefully foster new research.

Although some of the methods described here have been extended to filter animation sequences, we believe this is the area that provides biggest potential for further quality improvements. It would also be tempting to try to combine the advantages of *a priori* methods, like taking advantage of sheared structures in light fields, and *a posteriori* techniques, like the complex filter shapes. Methods for real-time rendering still lag the quality of off-line methods, and more effort should be spent on trying to close this gap. Finally, there is a variety of techniques and methodologies that remain largely unexplored in the context covered by this survey, including the use of data-driven and learning-based approaches, or general frameworks for sparse sampling and reconstruction based on work in compressive sensing and matrix completion. From a more theoretical perspective, an important open question is also whether lower bounds on the number of samples can formally be proved.

Acknowledgments

We would like to thank the following persons and institutes for making 3D objects and scenes publicly available: headus/Rezard for the KILLEROO model, the Stanford 3D Scanning Repository for the DRAGON

Table 1: Summary of the main characteristics of the *a posteriori* methods cited. In the "ASR" column, "S" stands for adaptive sampling, and "R" for adaptive reconstruction. Methods performing both ("SR") also support iteratively adjusting the sampling density and adding more samples. The "Metric" column indicates the error metric used to guide adaptive sampling or reconstruction. If it is empty the method does not adapt to any error estimate. In the "Reconstruction" column, an additional "*" identifies methods that leverage information contained in feature buffers. The last column "Acr." lists acronyms used in the comparison figures.

Method	ASR	Metric	Reconstr.	Acr.
Mitchell 1987 [9]	S	Contrast	uniform	-
Rushmeier and Ward 1994 [15]	R	Variance	splatting	-
Bolin and Meyer 1998 [13]	S	Perceptual	uniform	-
MccCool 1999 [16]	R	Variance	aniso. diff.*	-
Ramasubramanian et al. 1999 [14]	S	Perceptual	uniform	-
Farrugia and Péroche 2004 [49]	S	Perceptual	uniform	-
Xu and Pattanaik 2005 [26]	R	-	bilateral	-
Hachisuka et al. 2008 [19]	SR	Contrast	SS nearest neighbor	
Overbeck et al. 2009 [17]	SR	Contrast	wavelet	-
Dammertz et al. 2010 [21]	R	-	wavelet*	-
Bauszat et al. [37]	R	-	local regr.*	-
Rousselle et al. 2011 [18]	SR	MSE (bias, var.)	Gaussian	-
Sen and Darabi 2011, 2012 [2, 3]	R	Mutual inf.	bilateral*	RPF
Rousselle et al. 2012 [4]	SR	Variance	NL-means	NLM
Li et al. 2012 [31]	SR	MSE (SURE)	bilateral*	SURE
Kalantari et al. 2013 [1]	SR	Variance	gen. denoising	AMLD
Rousselle et al. 2013 [5]	SR	MSE (SURE)	bilateral*	RDFC
Delbracio et al. 2014 [27]	R	χ^2	NL-means	-
Moon et al. 2014 [6]	SR	MSE (bias, var.)	local regr.*	LWR
Kalantari et al. 2015 [7]	R	machine learning (MLP)	bilateral*, NL-means*	LBF

model, the Cornell University Program of Computer Graphics for the CORNELL BOX, Guillermo M. Leal Llaguno of Evolución Visual for the SAN MIGUEL scene, Andrew Kensler and the Utah 3D Animation Repository for the TOASTERS scene, Marko Dabrovic and Mihovil Odak for the SIBENIK CATHEDRAL, Anat Grynberg and Greg Ward for the CONFERENCE ROOM, and Jo Ann Elliott for the KITCHEN scene. P. Sen was funded by NSF grant 1342931. S. E. Yoon was supported in part by NRF-2013R1A1A2058052. M. Zwicker acknowledges support by the Swiss National Science foundation under grant no. 143886. Finally, we are grateful to the Monte Carlo rendering research community, in particular those working in adaptive sampling and reconstruction whose work we discuss here. They, too, have contributed to this course.

Appendix

Code for the following methods is available online (retrieved May 2015):

- Hachisuka et al. 2008 [19]:
  `http://www.ci.i.u-tokyo.ac.jp/~hachisuka/mdas.zip`
- Egan et al. 2009 [28]:
  `http://www.cs.columbia.edu/cg/mb/`
- Overbeck et al. 2009 [17]:
  `http://www1.cs.columbia.edu/~rso2102/AWR/`
- Rousselle et al. 2011 [18]:
  `http://www.cgg.unibe.ch/downloads/asr-auxiliary.zip/at_download/file`
- Sen and Darabi 2011, 2012 [2, 3]:
  `http://dx.doi.org/10.7919/F4MW2F28`
- Li et al. 2012 [31]:
  `http://www.cmlab.csie.ntu.edu.tw/project/sbf/`
- Rousselle et al. 2012 [4]:
  `http://www.cgg.unibe.ch/downloads/nlm-code-data.zip/at_download/file`
- Kalantari and Sen 2013 [1]:
  `http://dx.doi.org/10.7919/F47P8W9K`
- Moon et al. 2013 [38]:
  `http://sglab.kaist.ac.kr/VFL/`
- Rousselle et al. 2013 [5]:
  `http://www.cgg.unibe.ch/downloads/pg2013_code_data.zip/at_download/file`
- Moon et al. 2014 [6]:
  `http://sglab.kaist.ac.kr/WLR/`
- Kalantari et al. 2015 [7]:
  `http://dx.doi.org/10.7919/F4CC0XM4`

References

[1] Nima Khademi Kalantari and Pradeep Sen. Removing the noise in Monte Carlo rendering with general image denoising algorithms. *Computer Graphics Forum*, 32(2):93–102, 2013.

[2] Pradeep Sen and Soheil Darabi. Implementation of random parameter filtering. Technical Report EECE-TR-11-0004, University of New Mexico, September 2011.

[3] Pradeep Sen and Soheil Darabi. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph.*, 31(3):18:1–18:15, June 2012.

[4] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive rendering with non-local means filtering. *ACM Trans. Graph.*, 31(6):195:1–195:11, November 2012.

[5] Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. Robust denoising using feature and color information. *Computer Graphics Forum*, 32(7):121–130, 2013.

[6] Bochang Moon, Nathan Carr, and Sung-Eui Yoon. Adaptive rendering based on weighted local regression. *ACM Trans. Graph.*, 33(5):170:1–170:14, September 2014.

[7] Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. A machine learning approach for filtering monte carlo noise. *ACM Trans. Graph.*, 34(4), July 2015.

[8] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986.

[9] Don P. Mitchell. Generating antialiased images at low sampling densities. *SIGGRAPH Comput. Graph.*, 21(4):65–72, August 1987.

[10] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. *SIGGRAPH Comput. Graph.*, 23(3):281–288, July 1989.

[11] Baining Guo. Progressive radiance evaluation using directional coherence maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 255–266. ACM, 1998.

[12] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92, June 1988.

[13] Mark R. Bolin and Gary W. Meyer. A perceptually based adaptive sampling algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 299–309. ACM, 1998.

[14] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 73–82. ACM, 1999.

[15] Holly E. Rushmeier and Gregory J. Ward. Energy preserving non-linear filters. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 131–138. ACM, 1994.

[16] Michael D. McCool. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Trans. Graph.*, 18(2):171–194, April 1999.

[17] Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. Adaptive wavelet rendering. *ACM Trans. Graph.*, 28(5):140:1–140:12, December 2009.

[18] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.*, 30(6):159:1–159:12, December 2011.

[19] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):33:1–33:10, August 2008.

[20] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[21] Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. Edge-avoiding à-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*, HPG '10, pages 67–75. Eurographics Association, 2010.

[22] K.. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007.

[23] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[24] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, Dec 2006.

[25] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846, 1998.

[26] R. Xu and S.N. Pattanaik. A novel Monte Carlo noise reduction operator. *Computer Graphics and Applications, IEEE*, 25(2):31–35, 2005.

[27] Mauricio Delbracio, Pablo Musé, Antoni Buades, Julien Chauvier, Nicholas Phelps, and Jean-Michel Morel. Boosting Monte Carlo rendering by ray histogram fusion. *ACM Trans. Graph.*, 33(1):8:1–8:15, February 2014.

[28] Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph.*, 28(3):93:1–93:13, July 2009.

[29] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.*, 23(3):673–678, August 2004.

[30] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, 23(3):664–672, August 2004.

[31] Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.*, 31(6):194:1–194:9, November 2012.

[32] Hyosub Park, Bochang Moon, Soomin Kim, and Sung-Eui Yoon. P-RPF: Pixel-based random parameter filtering for Monte Carlo rendering. In *Computer-Aided Design and Computer Graphics (CAD/Graphics), 2013 International Conference on*, pages 123–130, Nov 2013.

[33] D. Van De Ville and M. Kocher. Sure-based non-local means. *Signal Processing Letters, IEEE*, 16(11):973–976, 2009.

[34] Soham Uday Mehta, JiaXian Yao, Ravi Ramamoorthi, and Fredo Durand. Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Trans. Graph.*, 33(4):57:1–57:12, July 2014.

[35] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[36] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[37] Pablo Bauszat, Martin Eisemann, and Marcus Magnor. Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum*, 30(4):1361–1368, 2011.

[38] Bochang Moon, Jong Yun Jun, JongHyeob Lee, Kunho Kim, Toshiya Hachisuka, and Sung-Eui Yoon. Robust image denoising using a virtual flash image for Monte Carlo ray tracing. *Computer Graphics Forum*, 32(1):139–151, 2013.

[39] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[40] Alexander Keller, Ken Dahm, and Nikolaus Binder. Path space filtering. In *ACM SIGGRAPH 2014 Talks*, SIGGRAPH '14, pages 68:1–68:1, 2014.

[41] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Trans. Graph.*, 31(6):191:1–191:10, November 2012.

[42] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012.

[43] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107, 2005.

[44] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional lightcuts. *ACM Trans. Graph.*, 25(3):1081–1088, 2006.

[45] Christopher DeCoro, Tim Weyrich, and Szymon Rusinkiewicz. Density-based outlier rejection in monte carlo rendering. *Computer Graphics Forum*, 29(7):2119–2125, 2010.

[46] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2909–2916, June 2014.

[47] P. Sen and S. Darabi. Compressive rendering: A rendering application of compressed sensing. *Visualization and Computer Graphics, IEEE Transactions on*, 17(4):487–499, April 2011.

[48] Felix Heide, Gordon Wetzstein, Ramesh Raskar, and Wolfgang Heidrich. Adaptive image synthesis for compressive displays. *ACM Trans. Graph.*, 32(4):132:1–132:12, July 2013.

[49] Jean-Philippe Farrugia and Bernard Péroche. A progressive rendering algorithm using an adaptive perceptually based image metric. *Computer Graphics Forum*, 23(3):605–614, September 2004.

[50] Mark Meyer and John Anderson. Statistical acceleration for animated global illumination. *ACM Trans. Graph.*, 25(3):1075–1080, July 2006.

[51] Luke Goddard. Silencing the noise on Elysium. In *ACM SIGGRAPH 2014 Talks*, SIGGRAPH '14, pages 38:1–38:1, New York, NY, USA, 2014. ACM.

Tentative Course Slides

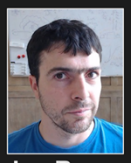
The next pages include a tentative version of the slides that will be used for this course.

Denoising Your Monte Carlo Renders:

Recent Advances in Image-Space Adaptive Sampling and Reconstruction

Pradeep Sen
UC Santa Barbara

Matthias Zwicker
University of Bern



Fabrice Rousselle
Disney Research

Sung-Eui Yoon
KAIST

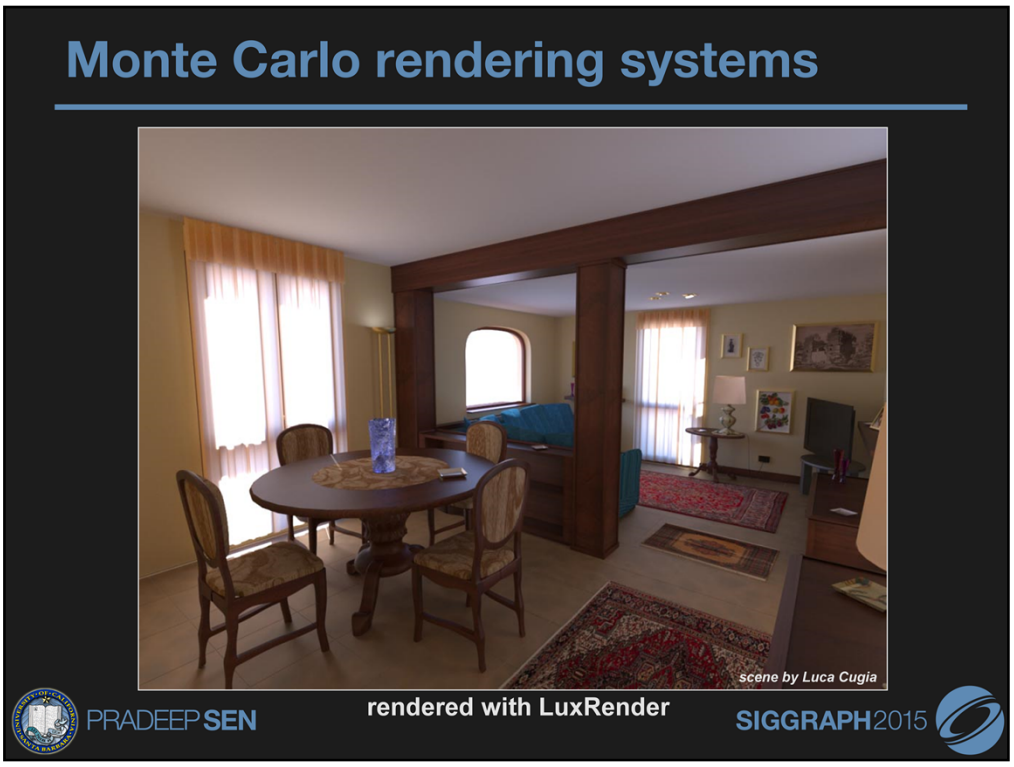
Nima Khademi Kalantari
UC Santa Barbara

SIGGRAPH2015

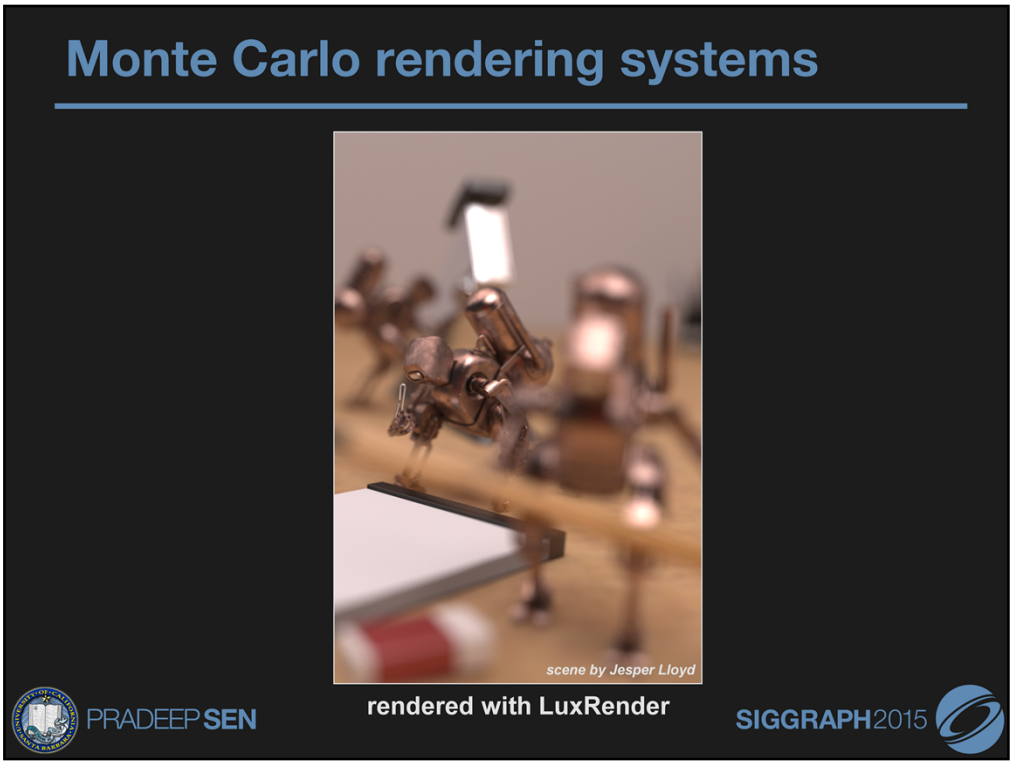
INTRODUCTION

scene by M. Leal Llaguno

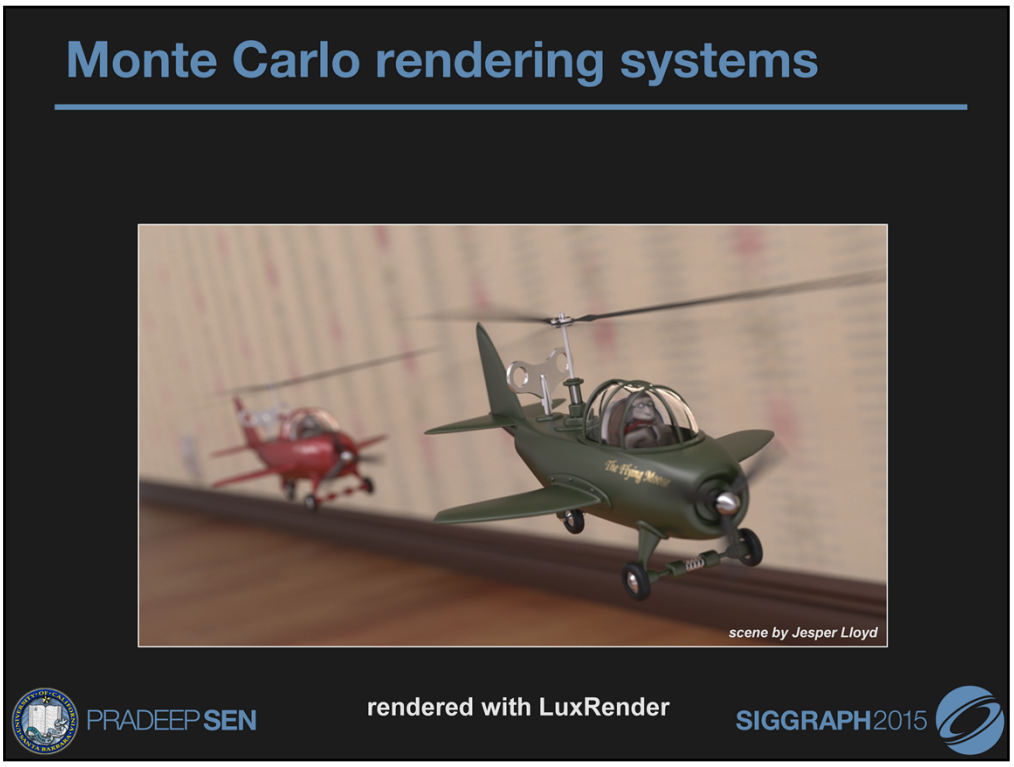
SIGGRAPH2015



Monte Carlo rendering systems are very powerful, and can produce visually stunning images.



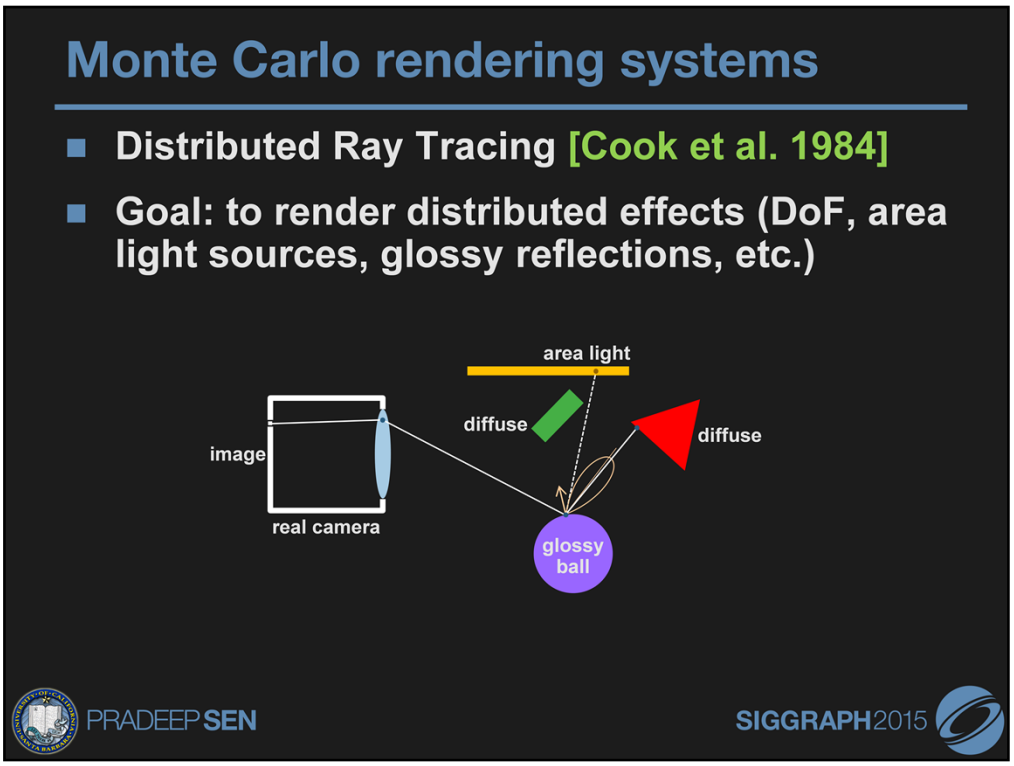
Monte Carlo rendering systems are very powerful, and can produce visually stunning images.



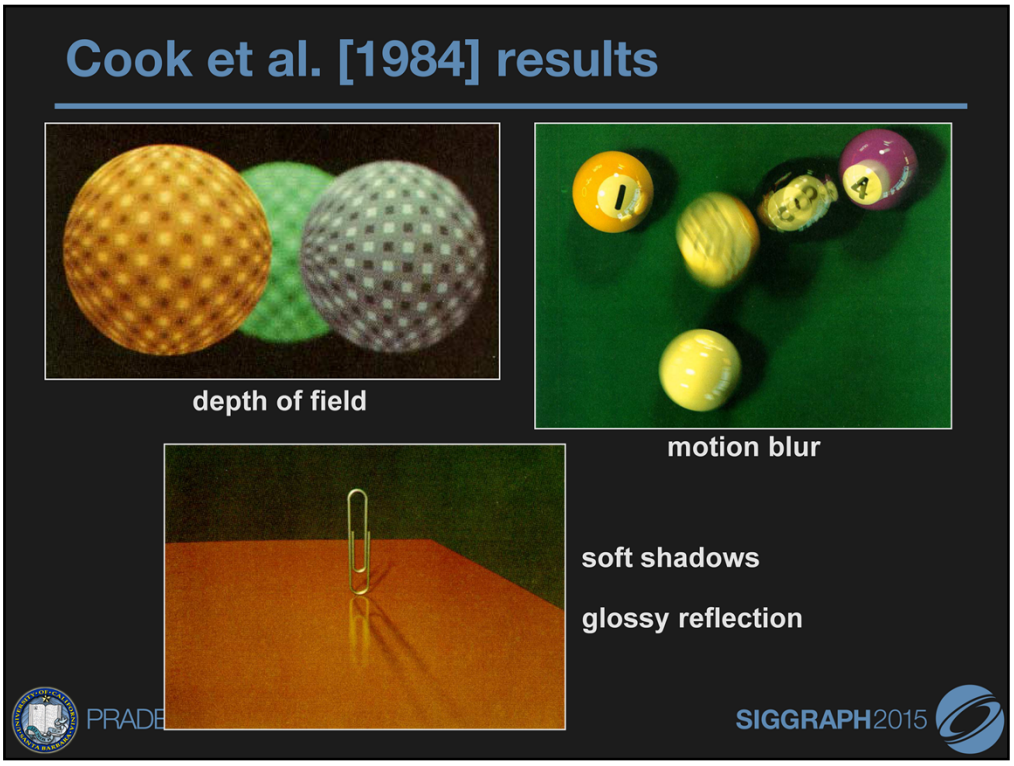
Monte Carlo rendering systems are very powerful, and can produce visually stunning images.

Monte Carlo rendering systems

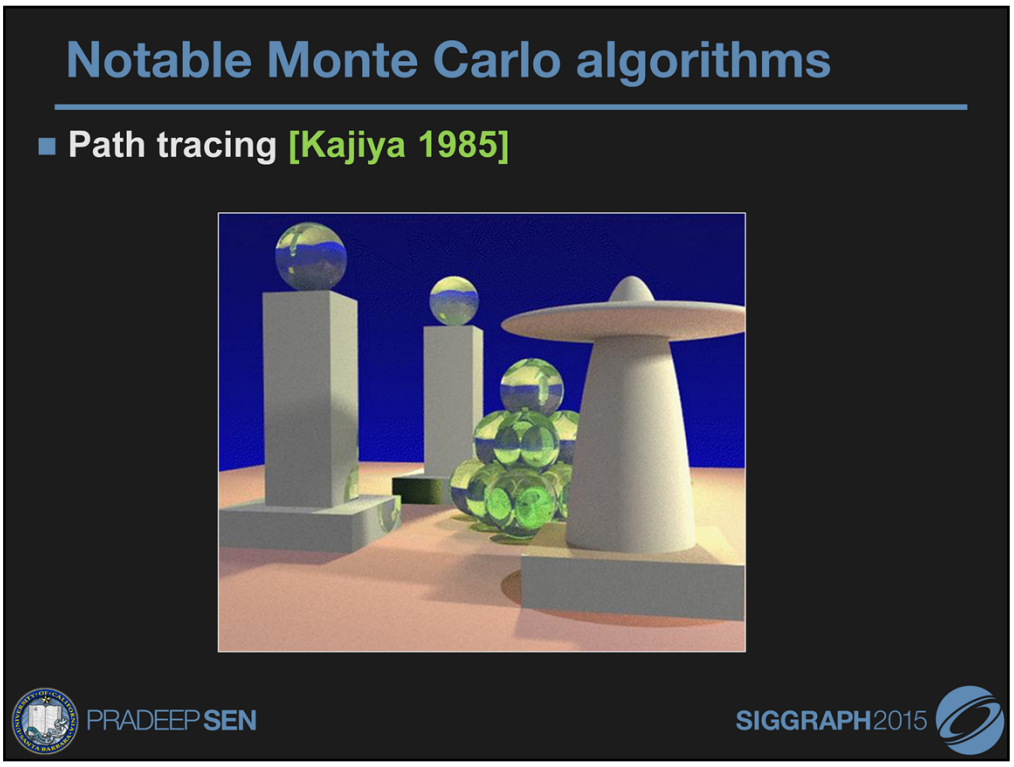
- Advantages:
  - + Physically-based results
  - + Flexible and general (all distributed effects)
  - + Little user interaction required
- Disadvantages:
  - – Potentially long rendering times
  - – Noisy images for short render times



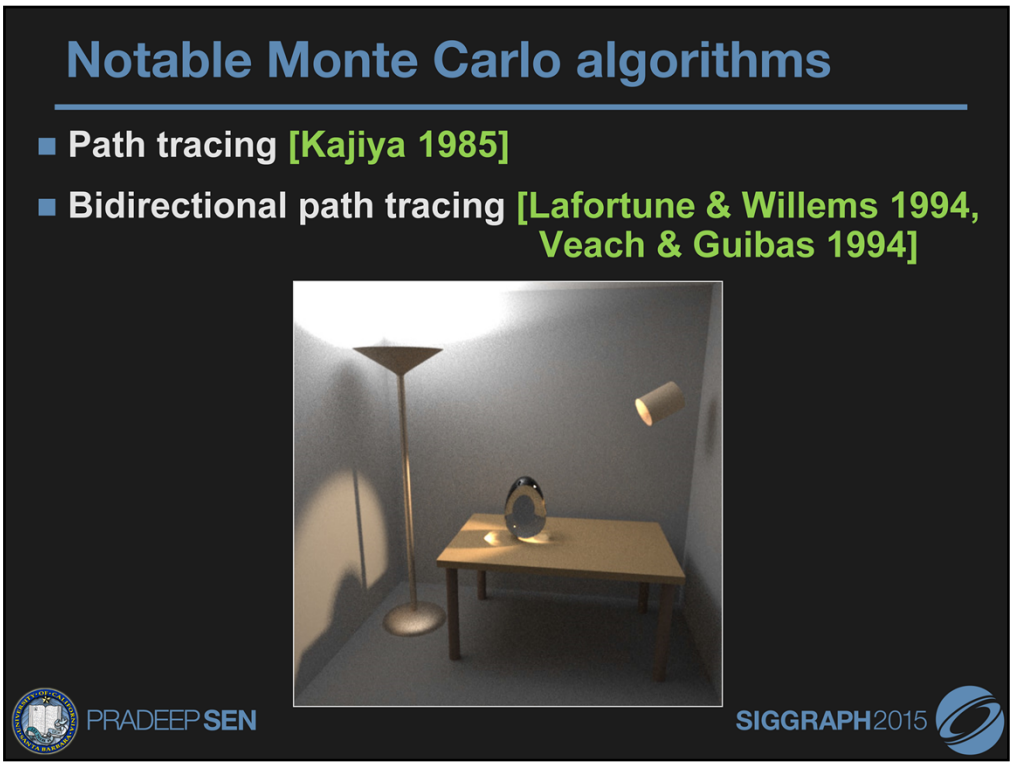
The first Monte Carlo rendering algorithm was proposed by Cook et al. in 1984.



They were able to demonstrate some very impressive distributed effects.



There has been considerable research in Monte Carlo algorithms for rendering, some of the notable algorithms shown here. This is by no means a complete list.



There has been considerable research in Monte Carlo algorithms for rendering, some of the notable algorithms shown here. This is by no means a complete list.

Notable Monte Carlo algorithms

- Path tracing [Kajiya 1985]
- Bidirectional path tracing [Lafortune & Willems 1994, Veach & Guibas 1994]
- Photon mapping [Jensen 1996]

PRADEEP SEN

SIGGRAPH 2015

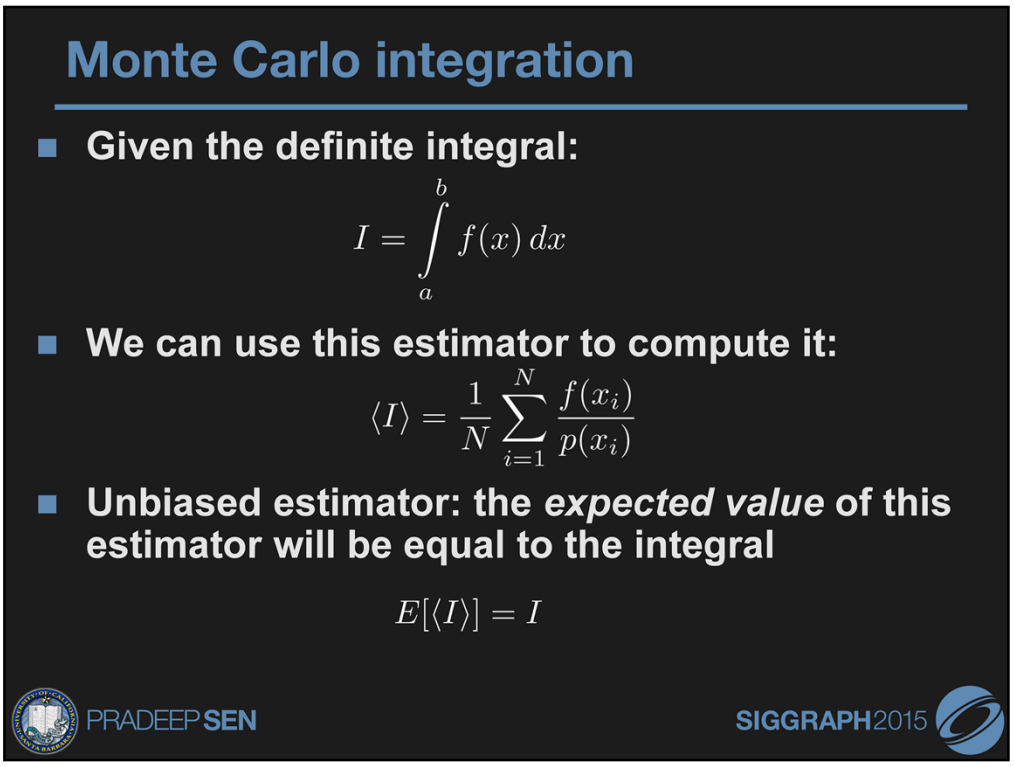
There has been considerable research in Monte Carlo algorithms for rendering, some of the notable algorithms shown here. This is by no means a complete list.

Notable Monte Carlo algorithms

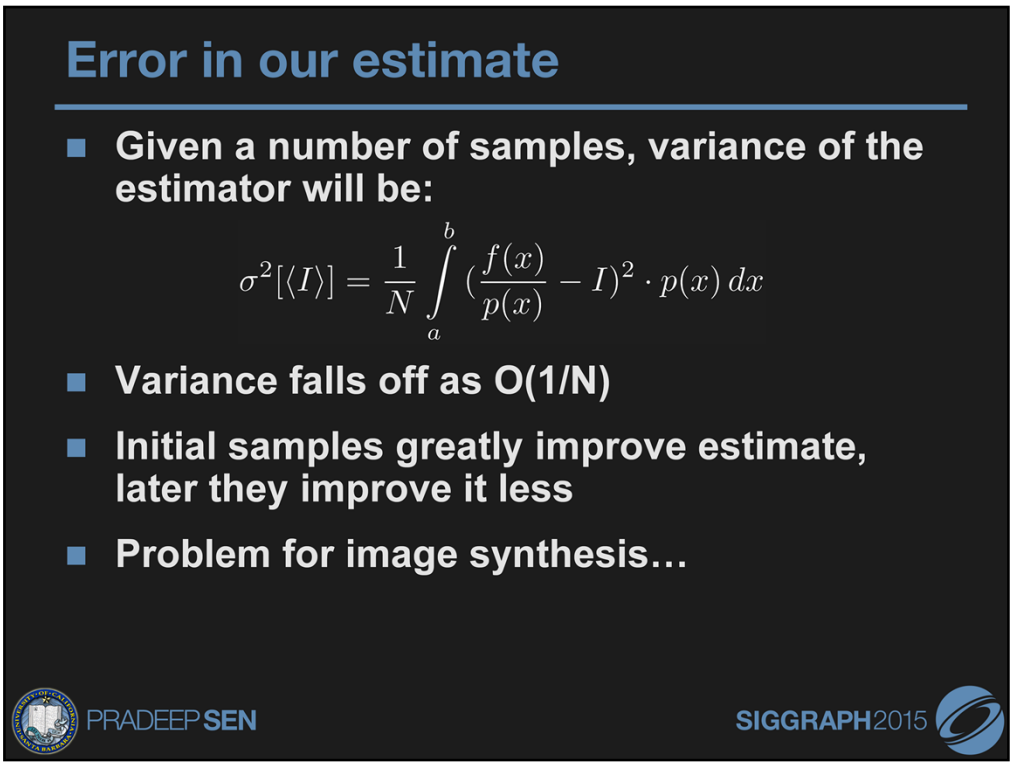
- Path tracing [Kajiya 1985]
- Bidirectional path tracing [Lafortune & Willems 1994, Veach & Guibas 1994]
- Photon mapping [Jensen 1996]
- Metropolis light transport [Veach & Guibas 1997]

PRAD SIGGRAPH2015

There has been considerable research in Monte Carlo algorithms for rendering, some of the notable algorithms shown here. This is by no means a complete list.



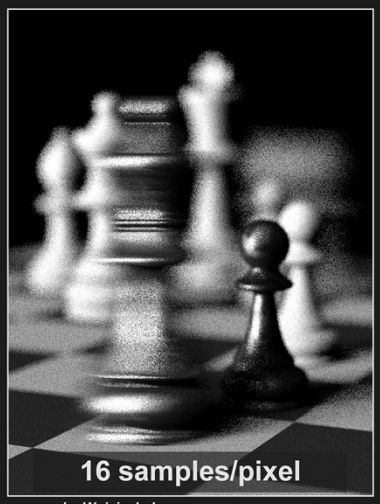
At their core, Monte Carlo algorithms use stochastic processes to calculate the integral of radiance over the different domains (over the aperture of the camera for depth of field, over the area light sources for soft shadows, over the BRDF for glossy reflections, over light paths for path tracing, etc.) We now present a basic overview of Monte Carlo integration.



However, this estimator will have inaccuracies, and the variance of the estimate is given by the equation shown.

The problem

- Variance appears as noise at low sample rates



scene by Wojciech Jarosz

The problem

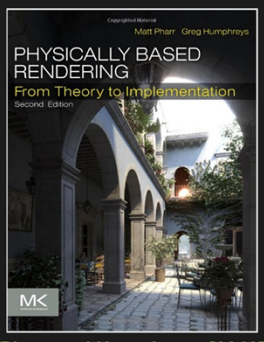
- More objectionable in animated sequences

8 samples/pixel (1.8 mins/frame)

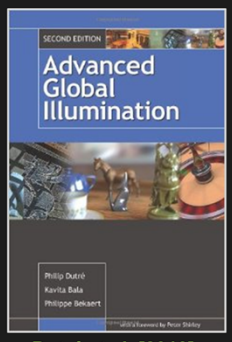
scene by M. Leal Llaguno

Previous work to reduce MC noise

- There has been over 30 years of research to address this problem
- Only provide cursory overview here, please refer to texts on the subject for more info…



Pharr and Humphreys [2010]



Dutré et al. [2010]

There has been over 30 years of research to address the problem of Monte Carlo noise, and we can only give a very cursory overview of previous work here. We refer interested readers to texts on the subject for a more in-depth overview.

Tracing rays faster

- Basic idea: compute samples faster so more samples can be used in estimate
- Sample innovations:
  - Faster intersection algorithms
    [Kay and Kajiya 1986] [Moller and Trumbore 1997] [Havel and Herout 2010]
  - Ray-traversal data structures
    [Glassner 1984] [Fujimoto et al. 1986] [Arvo and Kirk 1987] [Wald and Harven 2006] [Larsson and Akenine-Moller 2003] [Gunther et al. 2007] [Popov et al. 2009]
  - Memory coherent techniques
    [Pharr et al. 1997] [Wald et al. 2001, 2006] [Overbeck et al. 2008] [Garanzha and Loop 2010]
  - Parallel/GPU implementations
    [Purcell et al. 2002] [Foley and Sugerman 2005] [Aila and Laine 2009] [Parker et al. 2010] [van Antwerpen 2011]
  - Dedicated ray tracing hardware
    [Humphreys and Ananian 1996] [Schmittler et al. 2002,2004] [Caustic Graphics 2010] [Nah et al. 2011]

Better sampling techniques

- Basic idea: better sampling reduces noise
- Sample innovations:
  - Low discrepancy / Poisson disk
    [Dippé and Wold 1985] [Cook 1986] [Ulichney 1987] [Shirley 1991] [Kopf et al. 2006] [Wei 2008, 2010] [Lagae and Dutré 2008] [Schmaltz et al. 2012] [Chen et al. 2013]
  - Quasi Monte Carlo sampling patterns
    [Shirley 1991] [Keller 1996] [Keller et al. 2002]
  - Importance sampling
    [Whitted 1980] [Mitchell 1987, 1996] [Veach and Guibas 1995] [Clarberg et al. 2005], [Ramamoorthi et al. 2007] [Křivánek and Colbert 2008] [Soler et al. 2009]

Efficient MC integration algorithms

- Basic idea: reduce variance through better light path sampling
- Sample innovations:
  - Unbiased methods
    [Kajiya 1986] [Veach and Guibas 1994, 1997] [Lafortune and Willems 1993], [Jakob and Marschner 2012] [Lehtinen et al. 2013] [Hachisuka 2014]
  - Biased methods
    [Jensen 1996, 2001] [Keller 1997] [Walter et al. 2005, 2012] [Hachisuka et al. 2008, 2009] [Jarosz et al. 2008] [Knaus and Zwicker 2011] [Hachisuka and Jensen 2011] [Georgiev et al. 2012]

Exploiting properties of integrand

- Basic idea: integrand *f( )* has properties we can exploit to place or reuse samples
- Sample innovations:
  - Methods exploiting anisotropy
    [Hachisuka et al. 2008] [Egan et al. 2009, 2011] [Mehta et al. 2012, 2013, 2014] [Lehtinen et al. 2011, 2012] [Munkberg et al. 2014]
  - Methods exploiting transform domain
    [Chai et al. 2000] [Ramamoorthi and Hanrahan 2001] [Sloan et al. 2002, 2003], [Durand et al. 2005] [Soler et al. 2009] [Overbeck et al. 2009] [Sen and Darabi 2011] [Belcour et al. 2013]

Better models for specific scenes

- Basic idea: efficient algorithms for specific light transport effects and materials
- Sample innovations:
  - Subsurface scattering
    [Jensen 2001] [Donner and Jensen 2005] [Hašan et al. 2010] [Gkioulekas et al. 2013]
  - Hair rendering
    [Marschner et al. 2003] [Selle et al. 2008] [Jakob et al. 2009] [Xu et al. 2011]
  - Cloth rendering
    [Zhao et al. 2011, 2012] [Irawan and Marschner 2012] [Sadeghi et al. 2013]
  - Highly specular scenes
    [Jakob and Marschner 2012]

Why this course?

- Given all existing work, why this course?
  - Monte Carlo rendering of complex scenes still takes many hours
  - Algorithmic improvements have been offset by increasing scene complexity
  - Sometimes limits Monte Carlo rendering in production environments
  - We present approaches orthogonal to many of these previous methods
  - These new class of methods produce high quality results quickly

Observations

- After only a few minutes, we have a lot of information about the scene…

scene rendered in a few minutes (8 samples/pixel)

scene by M. Leal Llaguno

Observations

- Can we filter this to get a result as good as a reference image?

scene rendered in over 24 hrs (8,192 samples/pixel)

scene by M. Leal Llaguno

Observations

- In some cases, the Monte Carlo noise is very localized in the scene…

scene by Wojciech Jarosz

Observations

- Can we identify these regions and sample them more to reduce noise?

scene by Wojciech Jarosz

Adaptive sampling + reconstruction

- This suggests 2 kinds of noise reduction strategies, sometimes combined:
  1. Reconstruction (filtering) algorithms
     - Use information from renderer to remove MC noise directly
  2. Adaptive sampling algorithms
     - Use information from renderer to position new samples better to reduce noise
- Both methods have been explored in the past!

Previous work on filtering methods

- Denoising fully general MC distributed effects
  [Lee and Redner 1990] [Rushmeier and Ward 1994]
- Denoising specific distributed effects:
  - Illumination noise
    [Jensen & Christensen 1995] [McCool 1999] [Xu & Pattanaik 2005] [Meyer & Anderson 2006] [Segovia et al. 2006] [Laine et al. 2007] [Bauszat et al. 2011] [Egan et al. 2011]
  - Depth of field or motion blur noise
    [Egan et al. 2009] [Soler et al. 2009] [Chen et al. 2011]
  - Real-time illumination
    [Dammertz et al. 2010] [Mehta et al. 2012, 2013]
  - Real-time depth of field or motion blur noise
    [Shirley et al. 2011] [Munkberg et al. 2014]
  - Real-time illumination + depth of field noise
    [Mehta et al. 2014] [Bauszat et al. 2015]

Previous filtering result

[Xu and Pattanaik 2005]

input image

filtered result

Filtering approaches - issues

- Difficult distinguish between scene detail and noise – typically over-blur detail!

scene rendered in a few minutes (8 samples/pixel)

scene by M. Leal Llaguno

PRADEEP SEN

SIGGRAPH 2015

Filtering approaches - issues

- Difficult distinguish between scene detail and noise – typically over-blur detail!

scene rendered in over 24 hrs (8,192 samples/pixel)

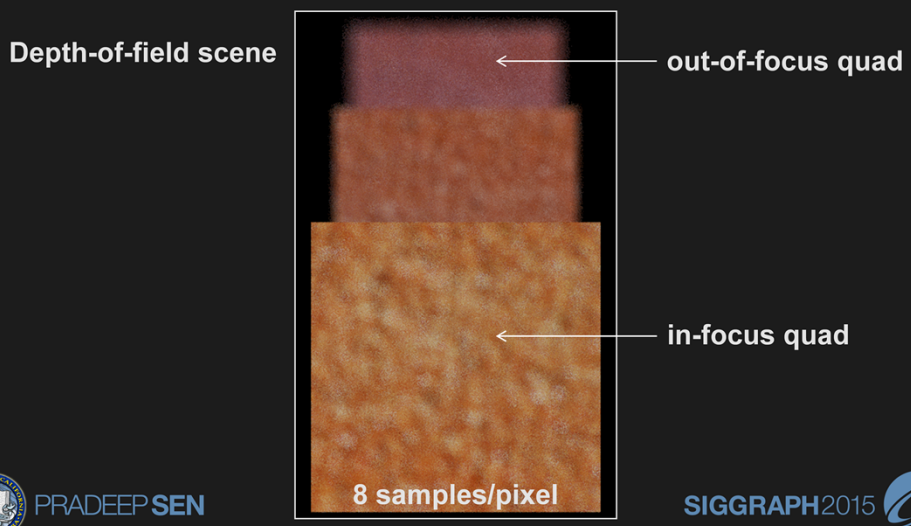
scene by M. Leal Llaguno

PRADEEP SEN

SIGGRAPH 2015

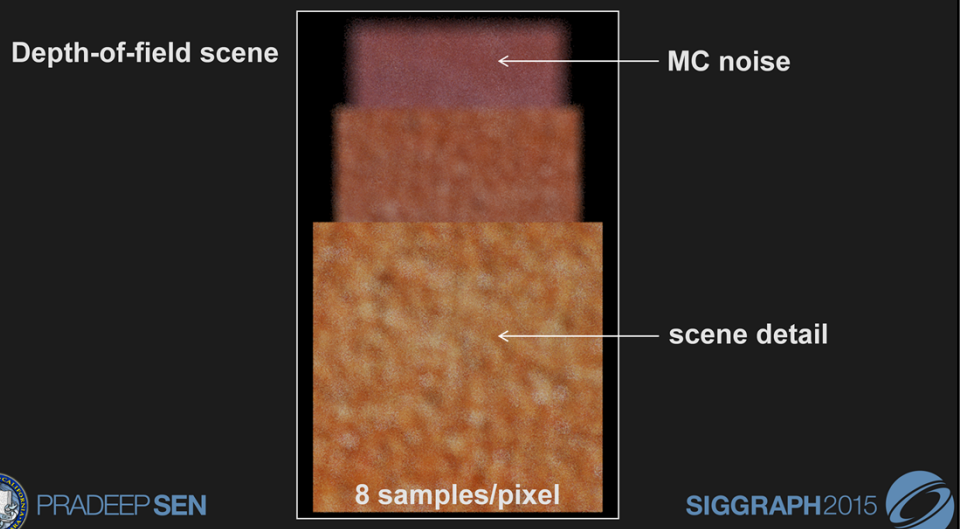
Filtering approaches - issues

- Difficult distinguish between scene detail and noise – typically over-blur detail!



Filtering approaches - issues

- Difficult distinguish between scene detail and noise – typically over-blur detail!



Filtering approaches - issues

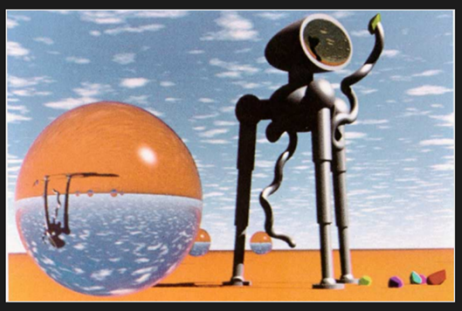
- Difficult distinguish between scene detail and noise – typically over-blur detail!
- This is why filtering algorithms have been largely ignored until recently…

Previous work on adaptive sampling

- Early on, researchers observed potential of adaptively sampling the scene

[Mitchell 1987]

sample map



rendered image

Previous work on adaptive sampling

- Early on, researchers observed potential of adaptively sampling the scene
- Large body of work using adaptive sampling to reduce Monte Carlo noise
  [Mitchell 1987, 1991] [Painter and Sloan 1989] [Bolin and Meyer 1998] [Guo 1998] [Bala et al. 2003] [Hachisuka et al. 2008] [Overbeck et al. 2009] [Soler et al. 2009] [Belcour et al. 2013]
- Typically use simple reconstruction kernel (box filter or sheared filter) at the back end
- In contrast, the work we present here combine adaptive sampling with more complex image-space filters (cross-bilateral, non-local means)

What makes this course different?

- Targeting high-quality, offline rendering
- Filtering as pure post-process to MC rendering
  - Using only information generated as a by-product of rendering
- Fully general – work for all distributed effects
- Work in image space, mostly independent of scene complexity

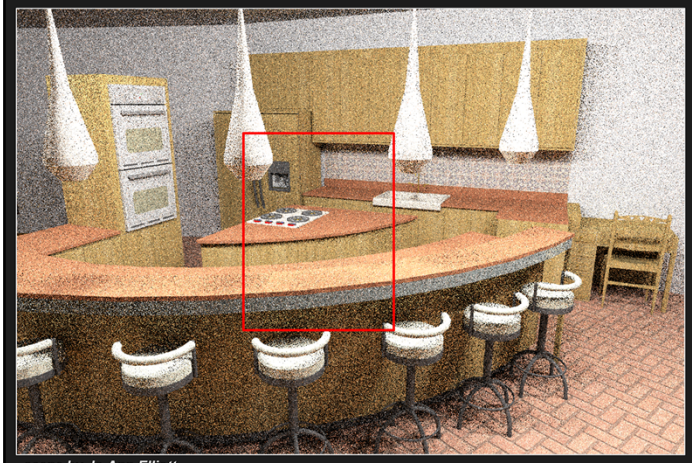
About these algorithms...

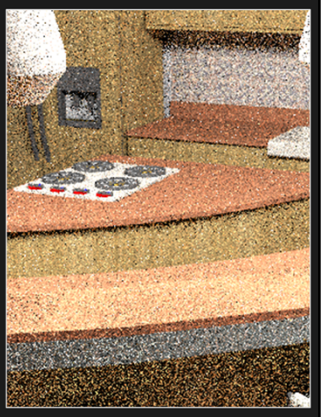
- Some combine adaptive sampling + filtering
  [Rousselle et al. 2011, 2012, 2013] [Kalantari and Sen 2013] [Moon et al. 2014]
- Others are entirely post-process filters
  [Sen and Darabi 2011, 2012] [Kalantari and Sen 2015]
- Biased algorithms which produce high-quality, noise-free results
- Source code for these algorithms is available

Sample result

Path traced scene

[Kalantari et al. 2015]





scene by Jo Ann Elliott

4 samples/pixel
(40.8 sec)

PRADEEP SEN

SIGGRAPH 2015

Sample result

Filtered result

[Kalantari et al. 2015]

scene by Jo Ann Elliott

4 samples/pixel
(48.9 sec)

using only post-process filter!

PRADEEP SEN

SIGGRAPH2015

Course outline

1. Introduction
2. MC denoising with general image denoising [Kalantari and Sen 2013]
3. Removing noise from random parameters [Sen and Darabi 2011, 2012]
4. Cross-bilateral and NL-means filtering with SURE-based error [Rousselle et al. 2012, 2013]
5. Local weighted regression [Moon et al. 2014]
6. Learning-based filtering [Kalantari et al. 2015]
7. Conclusions

Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms

Nima Khademi Kalantari and Pradeep Sen

University of California, Santa Barbara

Motivation

- **Image denoising techniques**

  Buades et al. [2005], Awate et al. [2006], Kervrann and Boulanger[2006], Kervrann and Boulanger [2008], Dabov et al. [2007], Elad and Aharon [2006], Aharon et al. [2006], Chatterjee and Milanfar [2009], Mairal et al. [2009], Joshi et al. [2009], Muresan and Parks [2003], Zhang et al. [2010], Liu et al. [2008], Chang et al. [2000], Portilla et al. [2003], Luisier et al. [2007], Donoho and Johnstone [1994], Angelino et al. [2010], Mairal et al. [2008], Brunet et al. [2009], Chatterjee and Milanfar [2012]

Motivation

- Image denoising techniques
  - ✓ High quality
  - ✓ Fast
  - ✗ Assume spatially invariant noise

Noisy

Motivation

- Image denoising techniques
  - ✓ High quality
  - ✓ Fast
  - ✘ Assume spatially invariant noise

Denoised

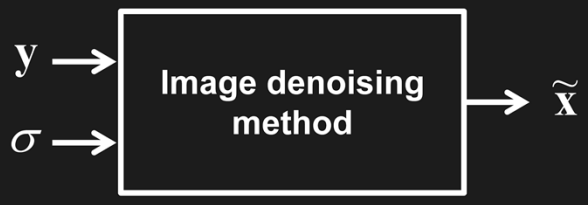
Our objective

- Apply these powerful techniques to remove the noise in Monte Carlo rendering

Background on image denoising

$$y_p = x_p + n$$

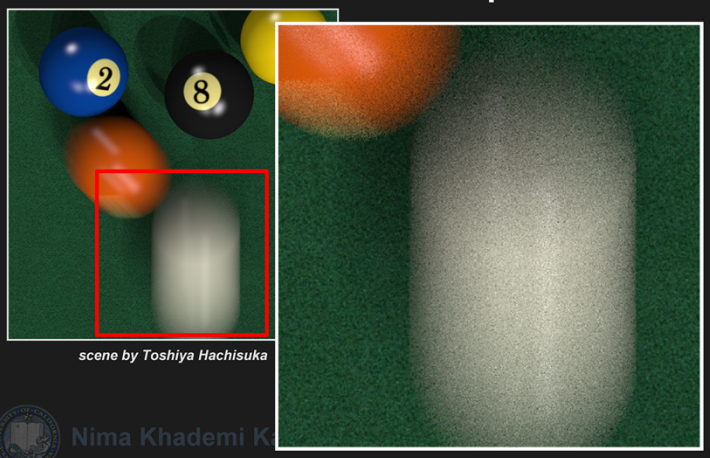
$$n \sim \mathrm{N}(0, \sigma^2)$$



Problem

- Assume constant variance for whole image
- Not good on rendered images

Input

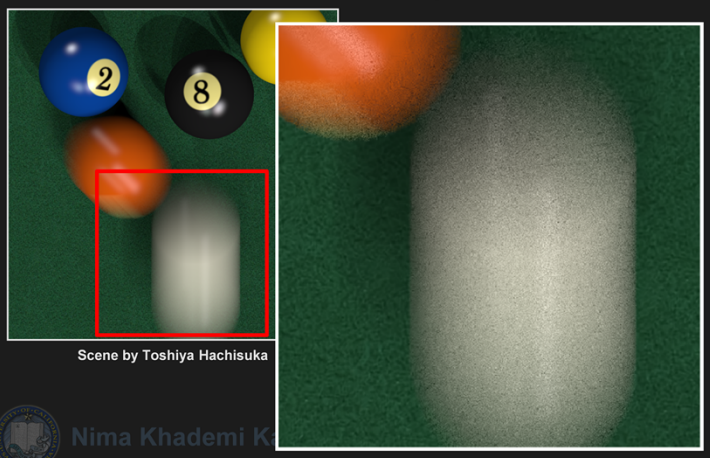


scene by Toshiya Hachisuka

Problem

- Assume constant variance for whole image
- Not good on rendered images

$$\sigma = 10$$

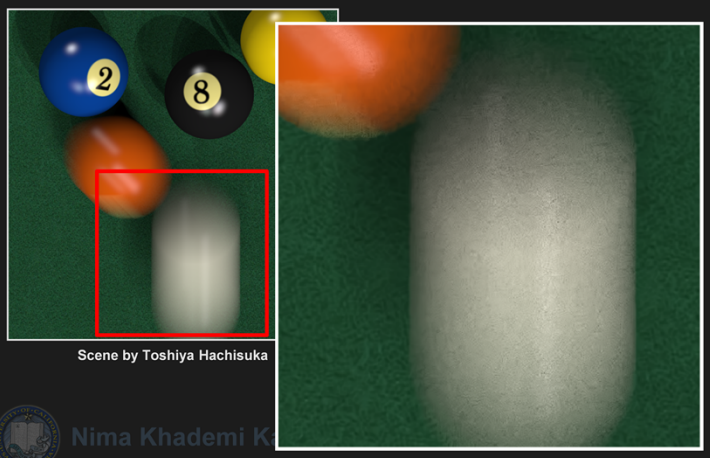


Scene by Toshiya Hachisuka

Problem

- Assume constant variance for whole image
- Not good on rendered images

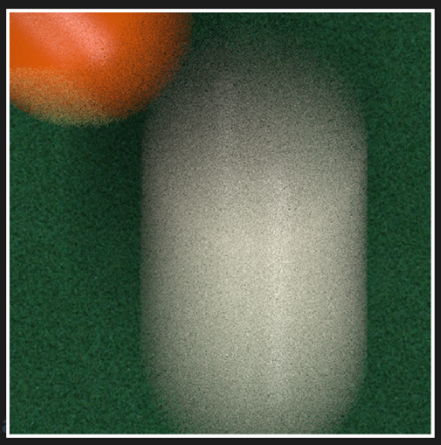
$$\sigma = 17$$



Scene by Toshiya Hachisuka

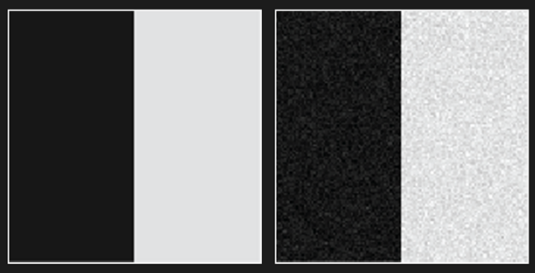
Our approach

- **Estimate noise level at each pixel**
- **Denoise each part of image using its noise level with standard denoising techniques**



Noise estimation problem

- It's tricky to separate edges from noise – both contribute to the pixel variance…



Original image | After adding noise

Proposed noise estimation

- **Median Absolute Deviation (MAD)**
  [Donoho and Johnstone 94]

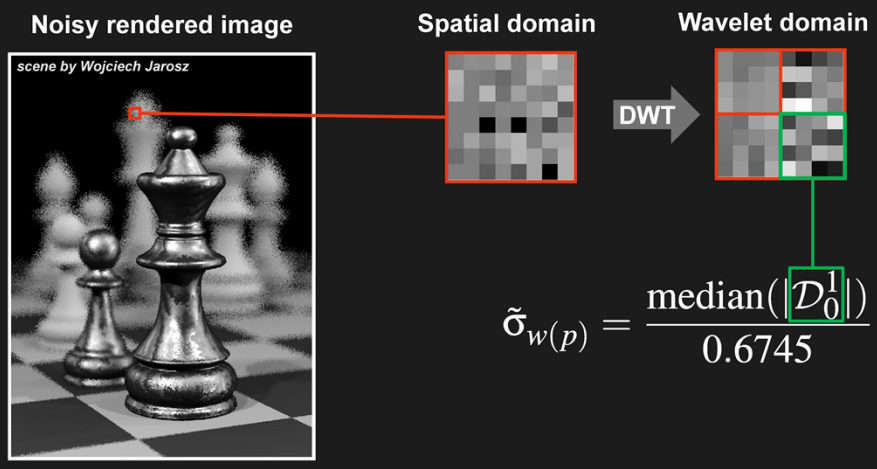
$$\tilde{\sigma}_{w(p)} = \frac{\text{median}(|\mathcal{D}_0^1|)}{0.6745}$$

$\mathcal{D}_0^1$ Diagonal detail coefficients of the finest level of wavelet transform

$\tilde{\sigma}_{w(p)}$ Estimated standard deviation of noise for a window around pixel p

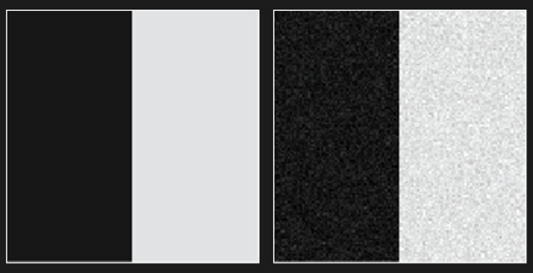
Assumption: noise is stationary inside a small window

Proposed noise estimation



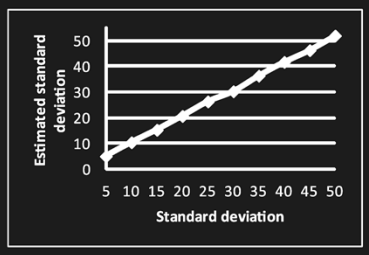
Proposed noise estimation

- It's tricky to separate edges from noise – both contribute to the pixel variance…
- The MAD estimator does a good job in separating the edges from the noise



Original image

After adding noise



Proposed noise estimation

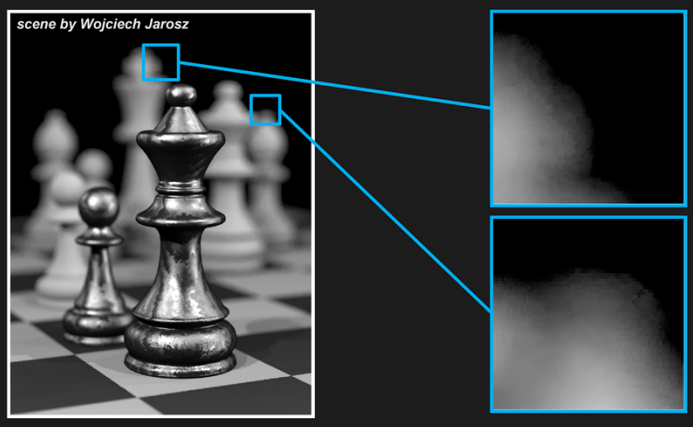
Noisy rendered image

Noise map

Denoised image

Proposed noise estimation

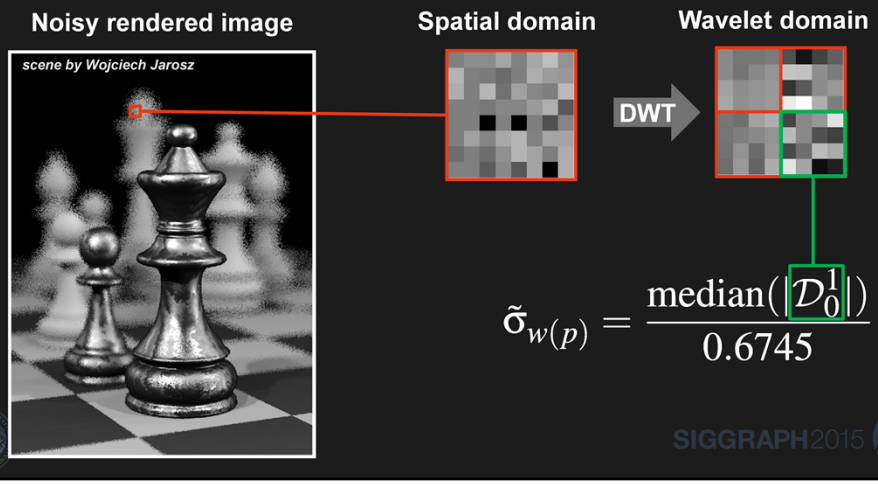
Denoised image



assumption of constant variance in small window violated

Proposed noise estimation

- $\tilde{\sigma}_{w(p)}$ is defined over a small window
- MAD assumption violated when noise variance changes inside the window



$$\tilde{\sigma}_{w(p)} = \frac{\text{median}(|\mathcal{D}_0^1|)}{0.6745}$$

Proposed noise estimation

- $\tilde{\sigma}_{w(p)}$ is defined over a small window
- MAD assumption violated when noise variance changes inside the window
- Need to localize the metric…

Proposed noise estimation

- Weight the MAD estimation by per pixel standard deviation of samples

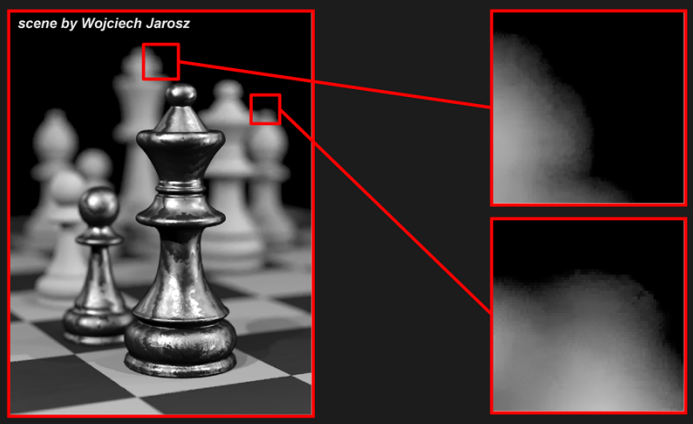
$$\tilde{\sigma}_p = \left(\frac{\tilde{\sigma}_{s(p)}}{\mu_{s(p)}}\right)^{1/4} \tilde{\sigma}_{w(p)}$$

- Dilate the map
- Normalize the map by its maximum and scale it by $g \cdot \max(\tilde{\sigma}_{w(p)})$

g – global parameter that defines smoothness of the results

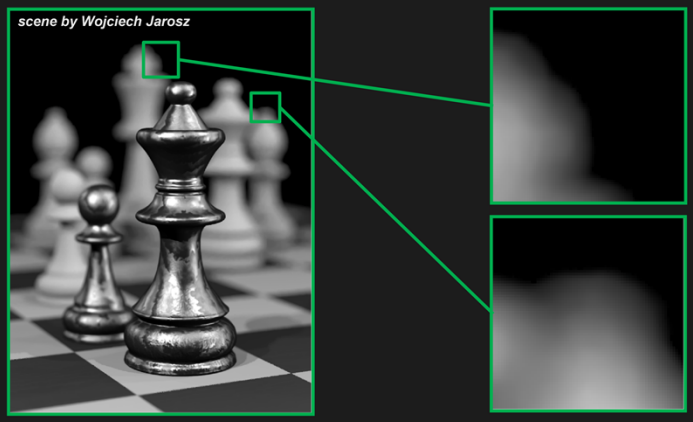
Proposed noise estimation

MAD only



Proposed noise estimation

Ours



Our approach

- Estimate noise level at each pixel
- Denoise each part of image using its noise level with standard denoising techniques

Multilevel denoising

Problem:

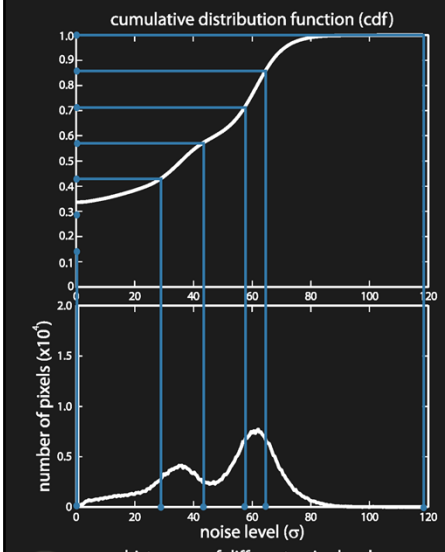
- Denoising each pixel using its $\tilde{\sigma}_p$ is not efficient

Our solution:

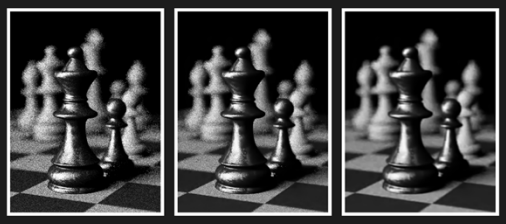
- Select a small subset of noise levels
- Denoise the rendered image a few times
- Pick the best pixel from denoised images

How to select the best set of noise levels to filter?

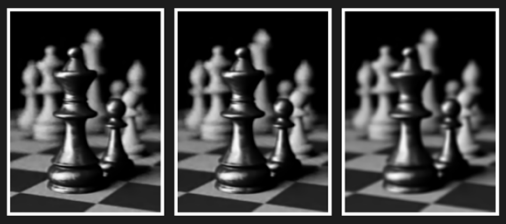
Multilevel denoising



scene by Wojciech Jarosz



σ = 0 σ = 28.73 σ = 43.16



σ = 57.46 σ = 66.44 σ = 118.24

Nima Khademi Kalantari

SIGGRAPH2015

Multilevel denoising

- For each pixel pick the two closest layers
- Interpolate the pixel value

$$I(p) = \alpha \cdot D_k(p) + (1-\alpha) \cdot D_{k-1}(p)$$

$$\alpha = (\tilde{\sigma}_p - \sigma(k-1))/(\sigma(k) - \sigma(k-1))$$

Adaptive sampling

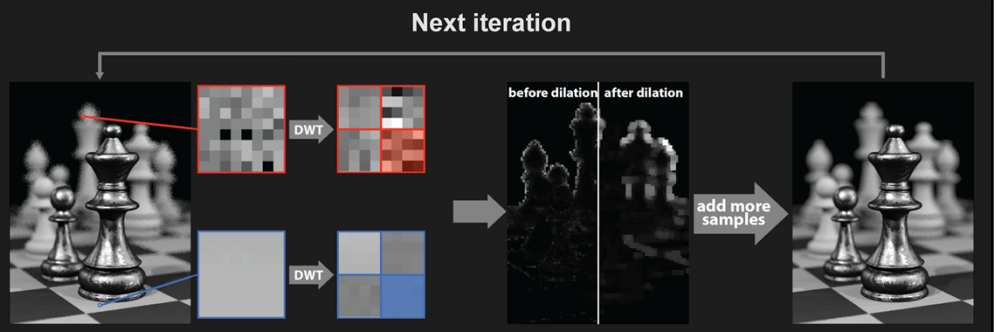
- Given we can estimate noise in the image, we can use this for adaptive sampling
- Proposed importance map based on MAD noise metric

$$\mathcal{M}_p = \sqrt{\gamma_{s(w(p))} \tilde{\sigma}^2_{w(p)}}$$

$\gamma_{s(w(p))}$ contrast metric by Hachisuka et al. [2008]

- Details in the paper

Adaptive sampling



Initial image
2 samples/pixel

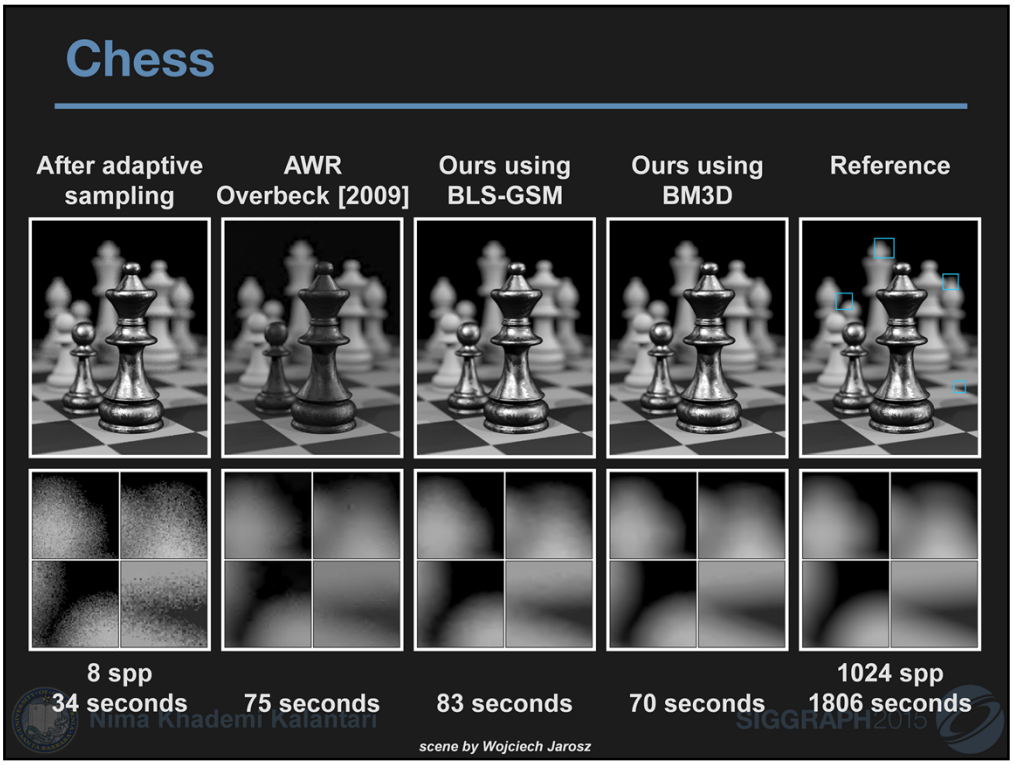
Importance map

Refined image
2+4 samples/pixel

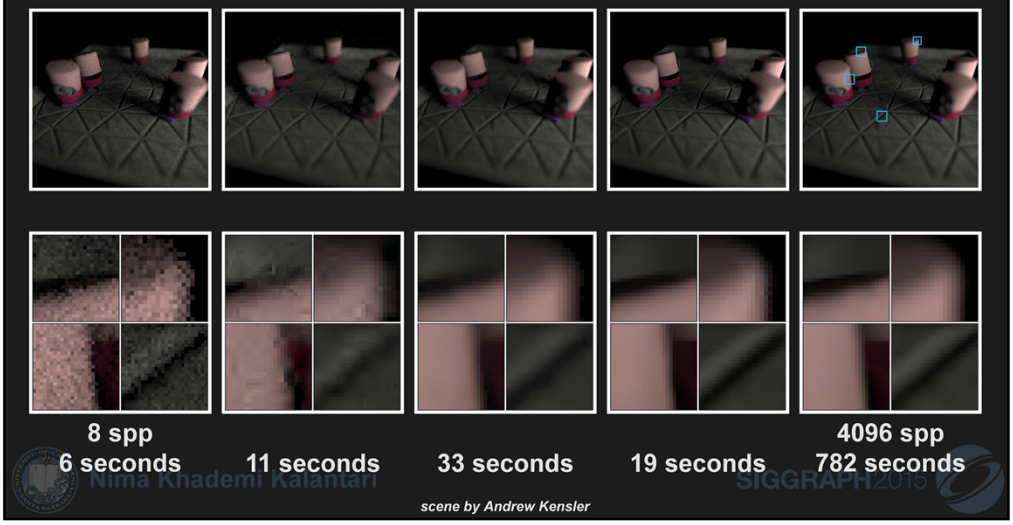
Nima Khademi Kalantari

SIGGRAPH2015

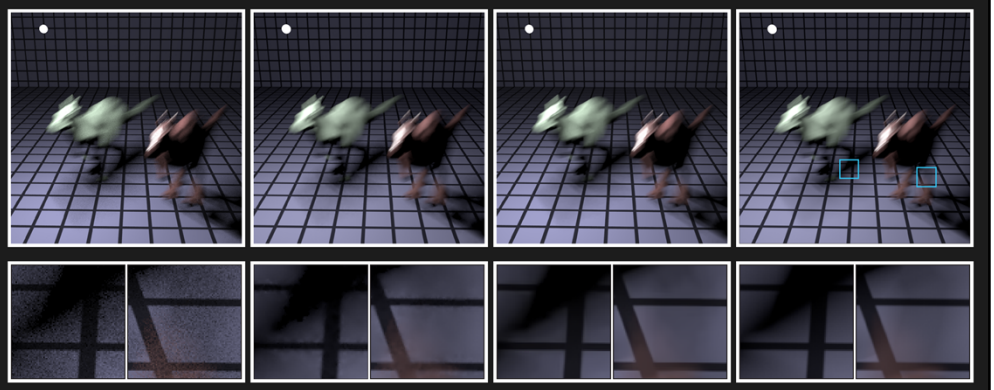
Results


Chess
After adaptive sampling
AWR Overbeck [2009]
Ours using BLS-GSM
Ours using BM3D
Reference
8 spp
34 seconds
75 seconds
83 seconds
70 seconds
1024 spp
1806 seconds
Nima Khademi Kalantari
SIGGRAPH2015
scene by Wojciech Jarosz

Toasters


After adaptive sampling
AWR Overbeck [2009]
Ours using BLS-GSM
Ours using BM3D
Reference
8 spp
6 seconds
11 seconds
33 seconds
19 seconds
4096 spp
782 seconds
Nima Khademi Kalantari
SIGGRAPH2015
scene by Andrew Kensler

Killeroos

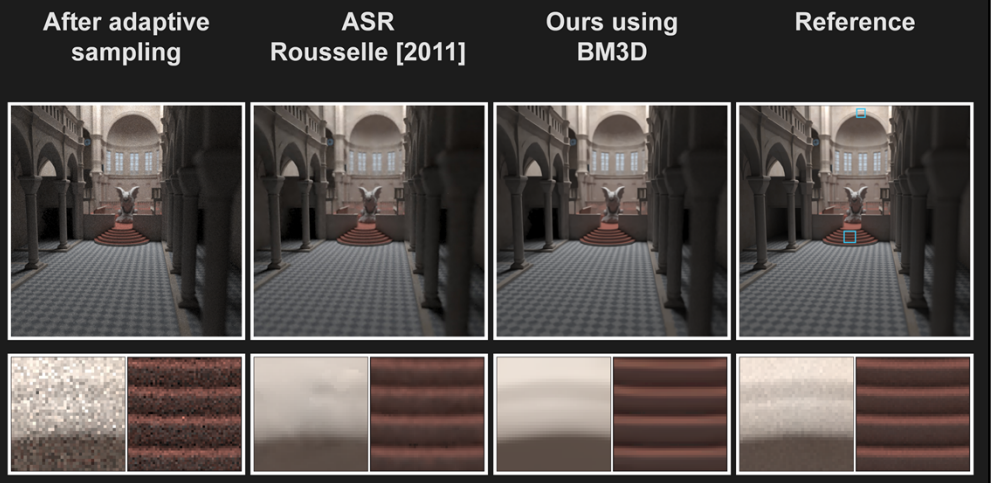


Nima Khademi Kalantari

SIGGRAPH2015

models by headus/Rezard

Sibenik



Nima Khademi Kalantari

SIGGRAPH2015

scene by Marko Dabrovic and Mihovil Odak

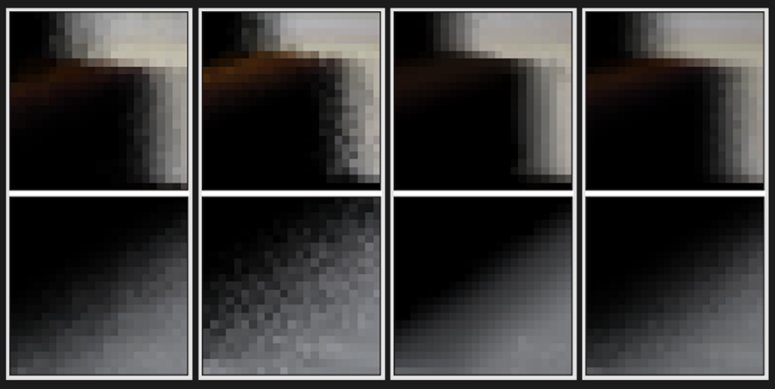
Moving Car

Ours using BM3D

Reference (256 spp)

scene by turbosquid.com user graphicdoom, Wikipedia user Jongleur100, and Kevin Egan

Moving Car (4 spp)



MDAS
Hachisuka [2008]

SFMB
Egan [2009]

Ours using
BM3D

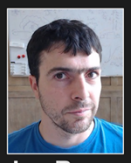
Reference
256 spp

Denoising Your Monte Carlo Renders:

Recent Advances in Image-Space Adaptive Sampling and Reconstruction

Pradeep Sen
UC Santa Barbara

Matthias Zwicker
University of Bern



Fabrice Rousselle
Disney Research

Sung-Eui Yoon
KAIST

Nima Khademi Kalantari
UC Santa Barbara

SIGGRAPH2015

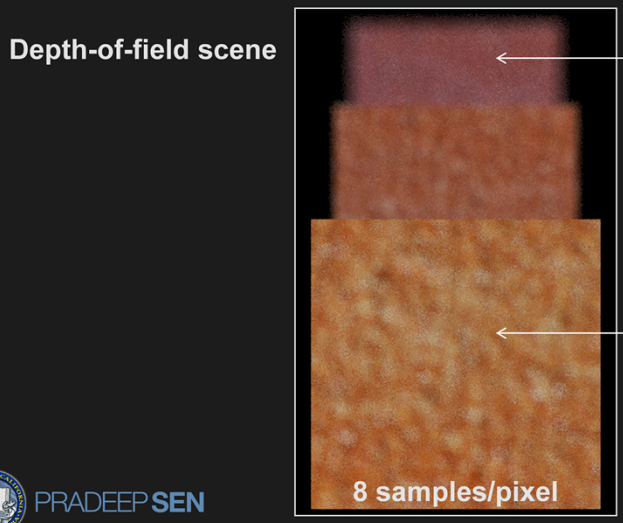
FILTERING NOISE FROM RANDOM PARAMETERS IN MC RENDERING

scene by Luca Cugia

SIGGRAPH2015

Problem with filtering approaches

- Difficult distinguish between scene detail and noise – typically over-blur detail!

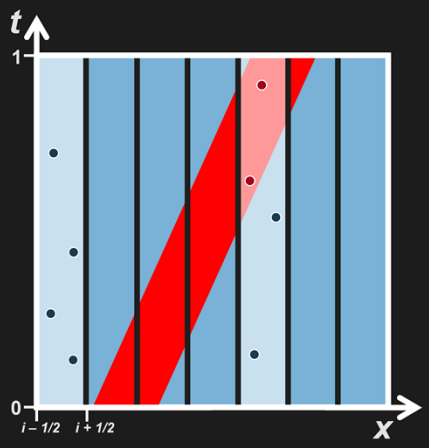


Our key observation

- Monte Carlo noise **occurs** when sample values are functions of the random parameters of the Monte Carlo system
- Desirable scene detail **is not** a function of these random parameters

$$I(i,j) = \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{0}^{1} \int_{-1}^{1} \int_{-1}^{1} \int_{0}^{1} \int_{0}^{1} f(x, y, l_1, l_2, u, v, t) \underbrace{dl_1\, dl_2\, du\, dv\, dt\, dx\, dy}_{\text{random parameters}}$$

Simple illustration

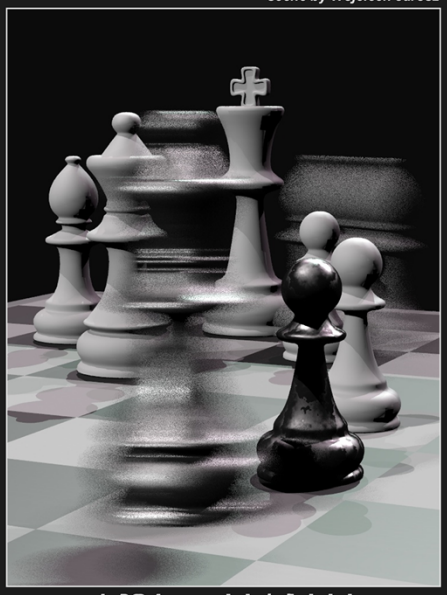


$$I(i,j) = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{0}^{1} f(x,t)\, dt\, dx$$

- Where $f(x,t)$ varies with respect to t, the random values for t will produce noise at the output
- Where $f(x,t)$ is constant with respect to t, high frequency content at the output is scene detail

Example

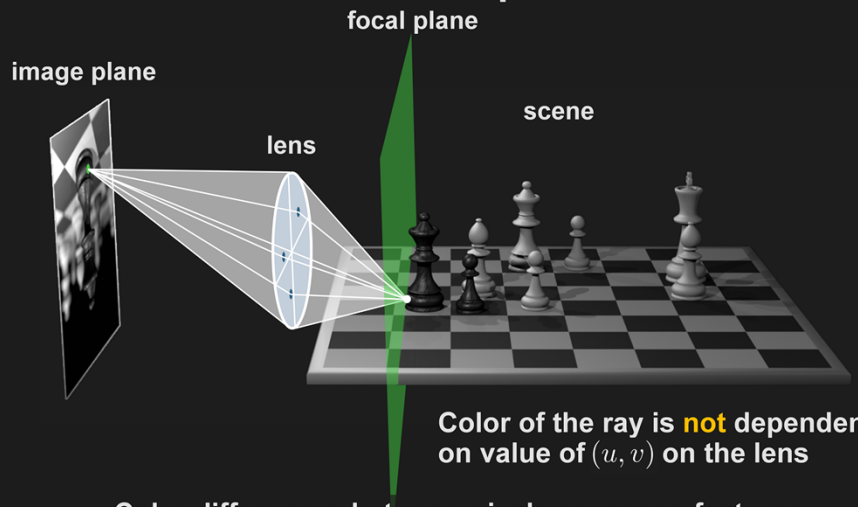
scene by Wojciech Jarosz



1,024 samples/pixel

Depth of Field (DoF)

- The same is true for depth of field effects

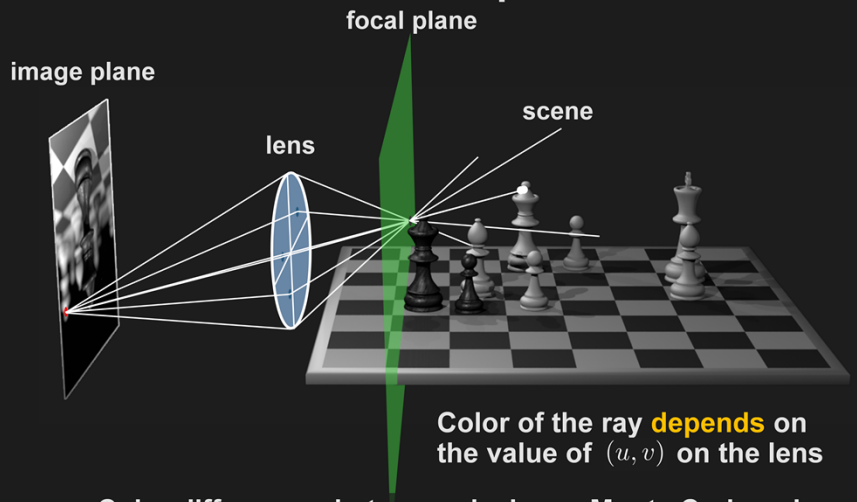


Color of the ray is **not** dependent on value of (u, v) on the lens

Color differences between pixels are scene features, **not MC noise**

Depth of Field (DoF)

- The same is true for depth of field effects

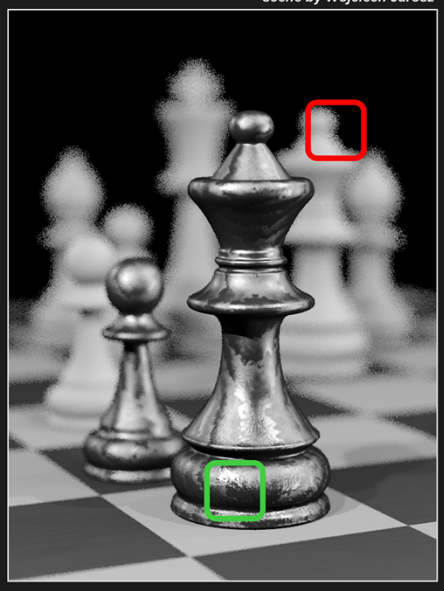


Color of the ray **depends** on the value of (u, v) on the lens

Color differences between pixels are Monte Carlo noise **not scene detail**

Depth of Field (DoF)

scene by Wojciech Jarosz



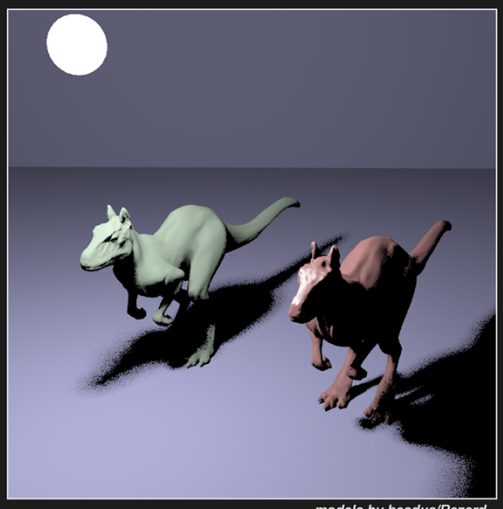
PRADEEP SEN

8 samples/pixel

SIGGRAPH2015

This works for other effects

- Area light sources



models by headus/Rezard

PRADEEP SEN

SIGGRAPH 2015

This works for other effects

- Area light sources
- Glossy reflections

Blender distribution scene

This works for other effects

- Area light sources
- Glossy reflections
- Path-tracing

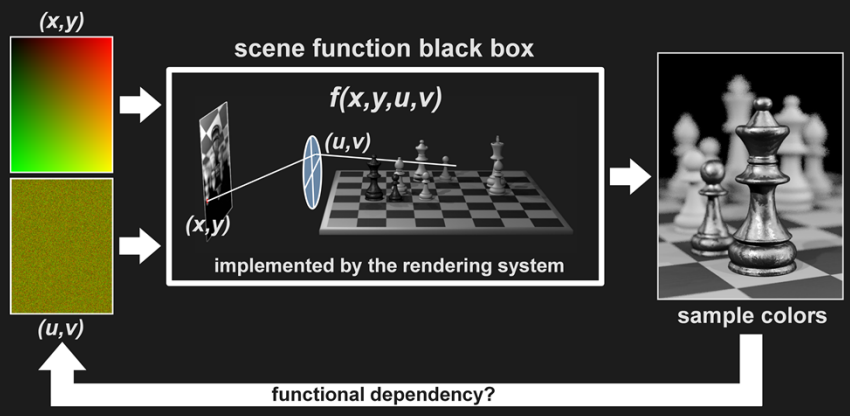
scene by Luca Cugia

Our key observations

- MC noise occurs **only** where sample values are functions of MC random parameters
- In fact, noise is directly caused by the random parameters (random numbers act as a noise source)
- If we identify these regions, we can design an adaptive filter to handle them correctly
- **The challenge:** how do we identify where this is happening in a general scene from just a few samples?

Our formulation

- To do this we treat the rendering system as a black box



We need a way to estimate this functional dependency

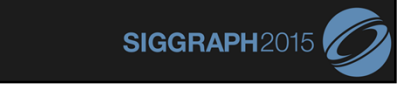
Estimating functional dependency

- Our rendering system can only take a small number of samples
- From these, we must determine if there exists a functional dependency between the outputs and inputs of our black box
- We turn to the field of information theory…

Mutual information

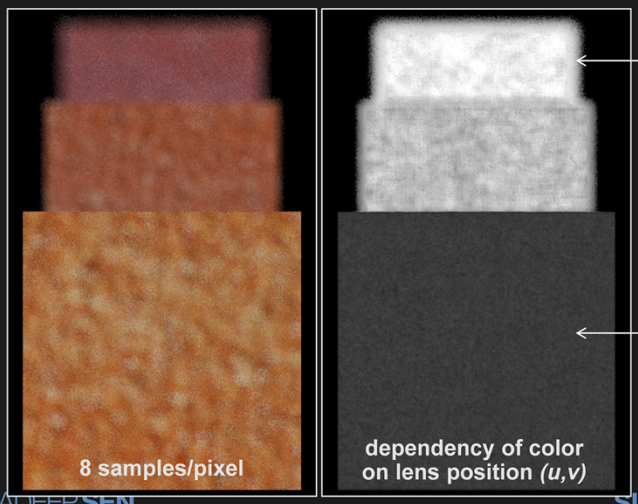
- The exact measure of dependence between two random variables
- It tells us how much information one random variable tells us about another
- Not exact measure of functional dependency, but works well in practice
- Computed as:

$$\mu(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$



Using mutual information

- Detects the dependence of the color on the lens position for our simple example

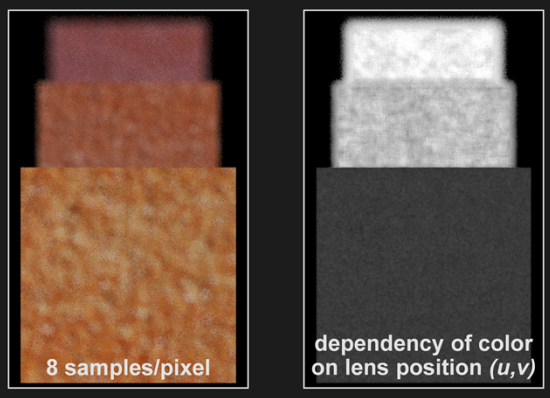


The next challenge

- Now that we have:
  - Observed that the noise is only in regions affected by random parameters
  - Proposed a way to identify these regions
- We need to develop an adaptive filter that removes only noise while preserving scene detail (e.g., edges, textures, etc.)
- We use a bilateral filter…

Bilateral filters for MC noise reduction

- Sample color differences due to MC noise are too large for bilateral filtering



- We can use our dependency to control importance of color in the bilateral filter

PRADEEP SEN

SIGGRAPH 2015

Our Random Parameter Filter (RPF)

- **Modify bilateral filter using mutual information to control impact of color channels**

$$w_{ij} = \exp\Big[-\frac{1}{2\sigma_{\mathbf{p}}^2}\sum_{1\le k\le 2}(\mathbf{p}_{i,k}-\mathbf{p}_{j,k})^2\Big] \times \exp\Big[-\frac{1}{2\sigma_{\mathbf{c}}^2}\sum_{1\le k\le 3}\alpha_k(\mathbf{c}_{i,k}-\mathbf{c}_{j,k})^2\Big] \times \exp\Big[-\frac{1}{2\sigma_{\mathbf{f}}^2}\sum_{1\le k\le m}\beta_k(\mathbf{f}_{i,k}-\mathbf{f}_{j,k})^2\Big]$$

- **If color is very dependent on random params, it is noise: set** $\alpha_k \to 0$
- **If color is not dependent on random params, set** $\alpha_k \to 1$

Filtering

- To filter each sample *i*, compute contribution weight of another sample *j* in the block:

$$w_{ij} = \exp\Big[-\frac{1}{2\sigma_{\mathbf{p}}^2}\sum_{1\le k\le 2}(\mathbf{p}_{i,k}-\mathbf{p}_{j,k})^2\Big]\times\exp\Big[-\frac{1}{2\sigma_{\mathbf{c}}^2}\sum_{1\le k\le 3}\alpha_k(\mathbf{c}_{i,k}-\mathbf{c}_{j,k})^2\Big]$$
$$\times\exp\Big[-\frac{1}{2\sigma_{\mathbf{f}}^2}\sum_{1\le k\le m}\beta_k(\mathbf{f}_{i,k}-\mathbf{f}_{j,k})^2\Big]$$

- Add its weighted color contribution to the color of the current sample:

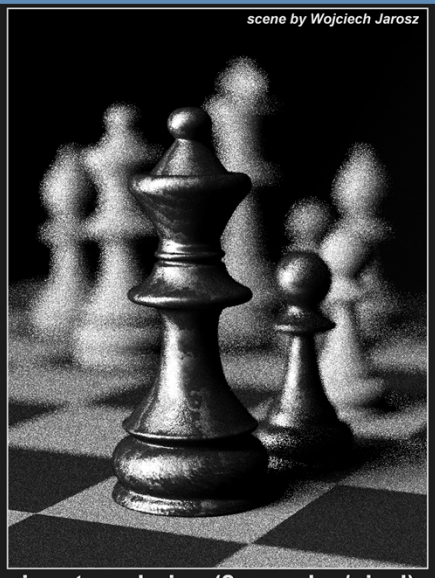
$$\mathbf{c}_i' = \frac{\sum_{j\in\text{block}} w_{ij}\mathbf{c}_j}{\sum_{j\in\text{block}} w_{ij}}$$

PRADEEP SEN

SIGGRAPH 2015

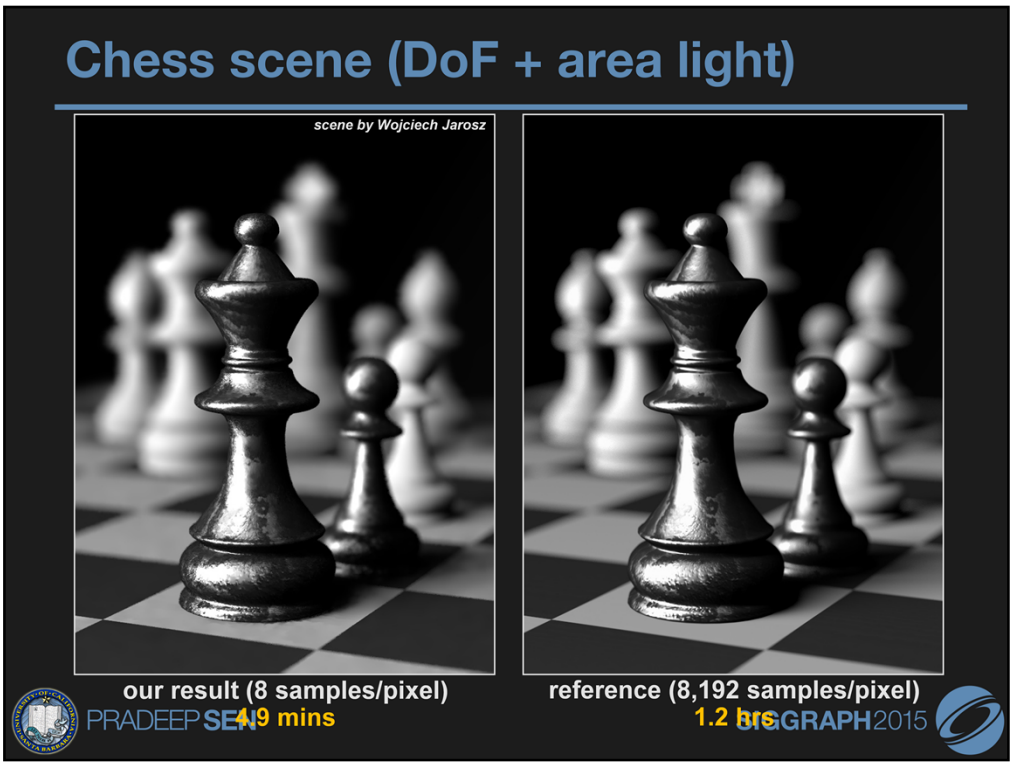
Results

Chess scene (DoF + area light)



input rendering (8 samples/pixel)
7.13 secs

reference (8,192 samples/pixel)
1.2 hrs


Chess scene (DoF + area light)
scene by Wojciech Jarosz
our result (8 samples/pixel)
4.9 mins
reference (8,192 samples/pixel)
1.2 hrs
PRADEEP SEN
SIGGRAPH 2015

Chess scene (DoF + area light)

MDAS (8 samples/pixel)
Hachisuka et al. [2008]

reference (8,192 samples/pixel)

Chess scene (DoF + area light)

Adaptive Wavelet Rendering (8 spp)
Overbeck et al. [2009]

reference (8,192 samples/pixel)

Chess scene (dof + area light)

À-Trous filter (8 samples/pixel)
Dammertz et al. [2010]

reference (8,192 samples/pixel)

Persian Room scene (path-tracing)

scene by Luca Cugia

input rendering (8 samples/pixel)
1.3 mins

reference (8,192 samples/pixel)
17 hours

PRADEEP SEN

SIGGRAPH2015

Persian Room scene (path-tracing)

scene by Luca Cugia

our result (8 samples/pixel)
3.75 mins

reference (8,192 samples/pixel)
17 hours

PRADEEP SEN

SIGGRAPH2015

Persian Room scene (path-tracing)

scene by Luca Cugia

equal-time rendering (32 samples/pixel)

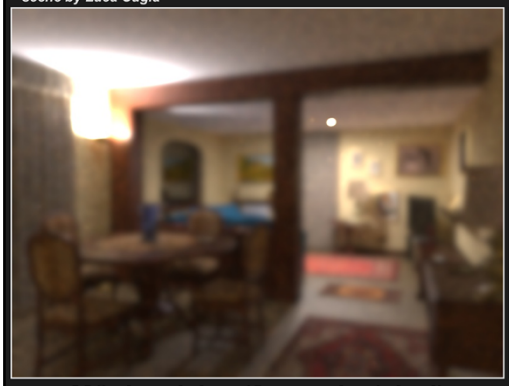
reference (8,192 samples/pixel)

PRADEEP SEN

SIGGRAPH 2015

Persian Room scene (path-tracing)

scene by Luca Cugia



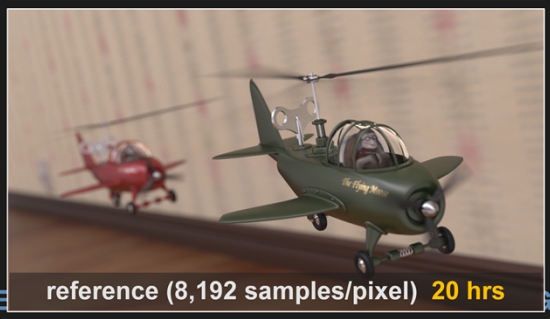
MC denoising (8 samples/pixel)
Xu and Pattanaik [2005]

reference (8,192 samples/pixel)

PRADEEP SEN

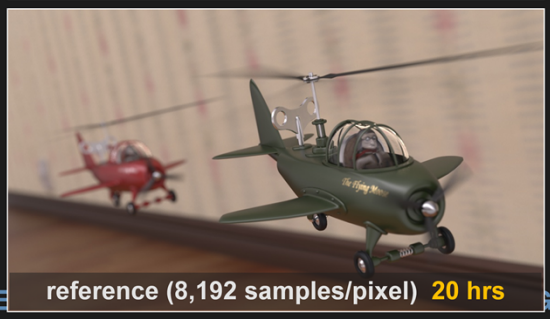
SIGGRAPH 2015

Toy Gyro scene



Toy Gyro scene

our result (8 samples/pixel) **15.9 mins**

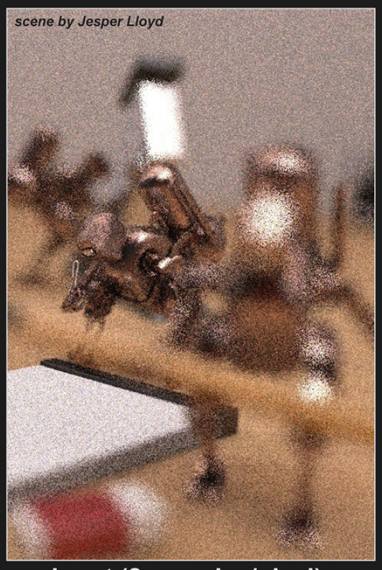


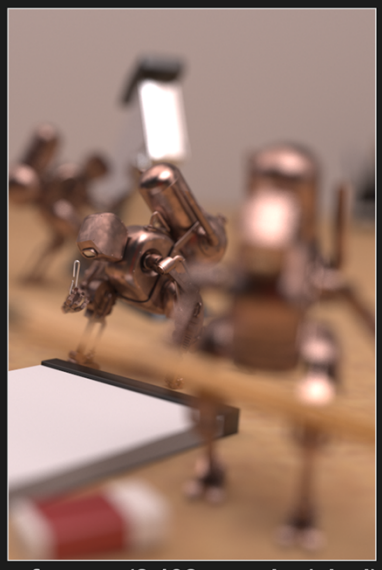
reference (8,192 samples/pixel) **20 hrs**

PRADEEP SE

GRAPH2015

Robots scene

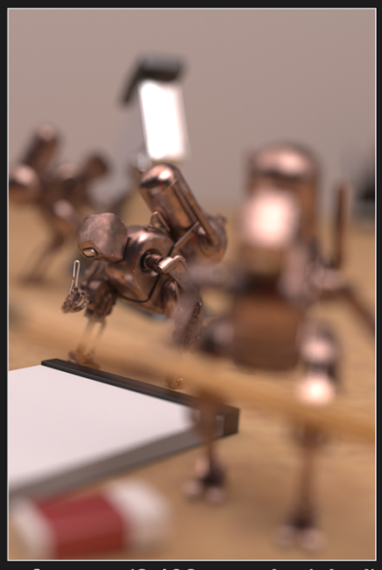




input (8 samples/pixel)
15 secs

reference (8,192 samples/pixel)
4 hours

Robots scene



our result (8 samples/pixel)
17 mins

reference (8,192 samples/pixel)
4 hours

San Miguel scene

San Miguel scene

PRADEEP S

GRAPH2015

San Miguel scene

San Miguel scene

San Miguel scene video (1920 x 1080)

scene by M. Leal Llaguno

input (8 samples/pixel)
1.8 mins/frame

San Miguel scene video (1920 x 1080)

scene by M. Leal Llaguno

our result (8 samples/pixel)
14.8 mins/frame

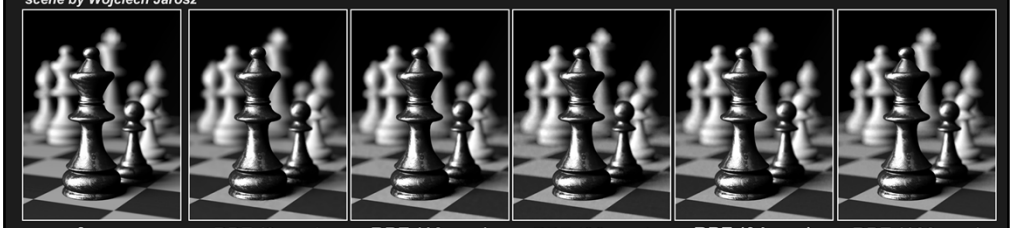
[reference
24.3 hrs/frame]

PRADEEP SEN

SIGGRAPH 2015

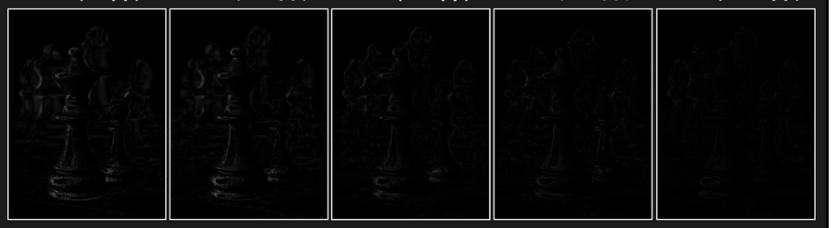
Quality as a function of sampling rate

scene by Wojciech Jarosz



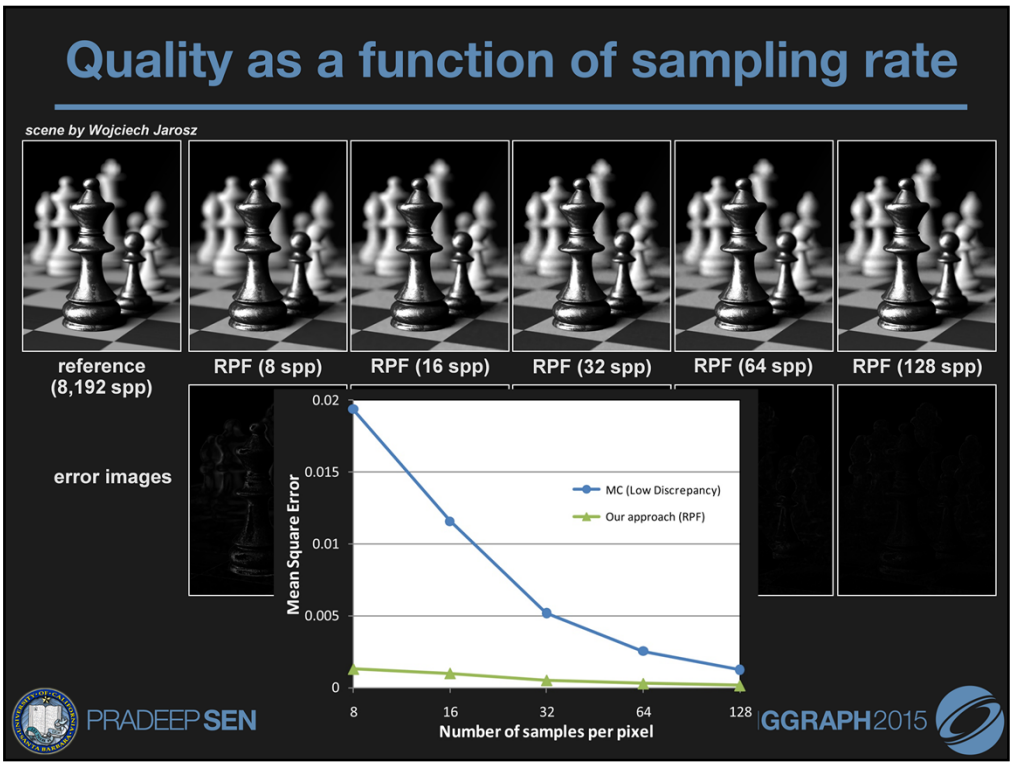
reference (8,192 spp) | RPF (8 spp) | RPF (16 spp) | RPF (32 spp) | RPF (64 spp) | RPF (128 spp)

error images

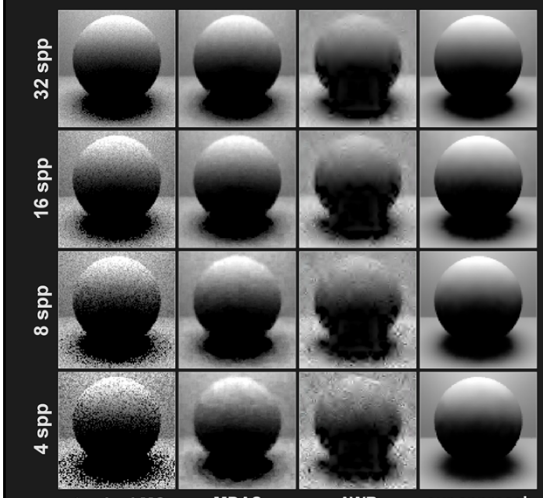


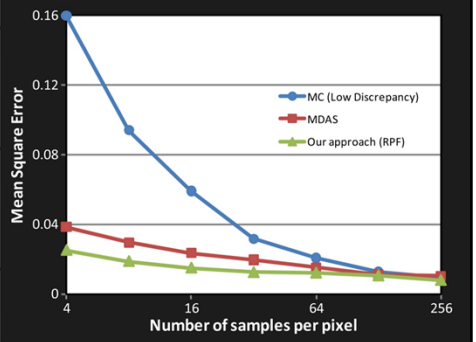
PRADEEP SEN

SIGGRAPH2015


Quality as a function of sampling rate
scene by Wojciech Jarosz
reference (8,192 spp)
RPF (8 spp)
RPF (16 spp)
RPF (32 spp)
RPF (64 spp)
RPF (128 spp)
error images
Mean Square Error
0.02
0.015
0.01
0.005
0
MC (Low Discrepancy)
Our approach (RPF)
8
16
32
64
128
Number of samples per pixel
PRADEEP SEN
SIGGRAPH2015

Quality as a function of sampling rate





PRADEEP SEN

SIGGRAPH 2015

Reflections, 5 years later

- RPF still provides reasonable filter quality
- Better than most methods at separating noise from scene detail
- Filter is sample based (not per-pixel), so scales poorly to higher sampling rates
- This makes it slower than later methods
- This problem addressed in:
  Park et al. 2013 “P-RPF: Pixel-based Random Parameter Filtering for Monte Carlo Rendering”
- But more work to be done!

The papers

- P. Sen and S. Darabi, “On Filtering the Noise from the Random Parameters in Monte Carlo Rendering,” *ACM Transactions on Graphics,* Vol. 31, No. 3, May 2012
  (presented at SIGGRAPH 2012)

- P. Sen and S. Darabi, “Implementation of Random Parameter Filtering,” University of New Mexico Tech. Report ECE-TR-11-0004, Sep. 2011

Source code available!

- **RPF code available here:**

  http://dx.doi.org/10.7919/F4MW2F28

Acknowledgments

- **Student:**

  Soheil Darabi

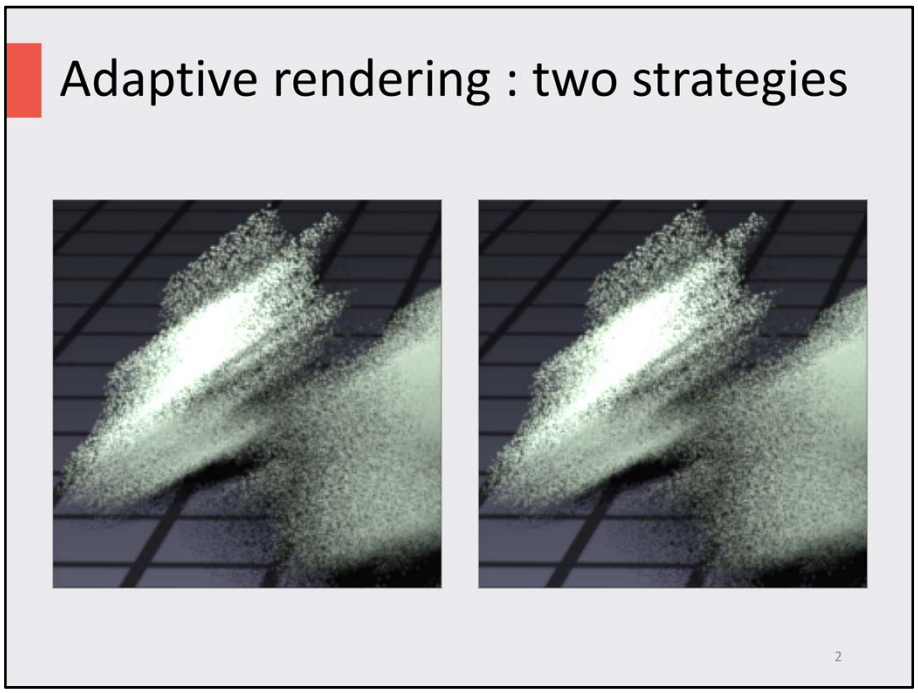
- **Funding:**

  National Science Foundation IIS 08-45396

Robust Adaptive Rendering using Joint NL-Means Filtering

Fabrice Rousselle

fabrice.rousselle@disneyresearch.com



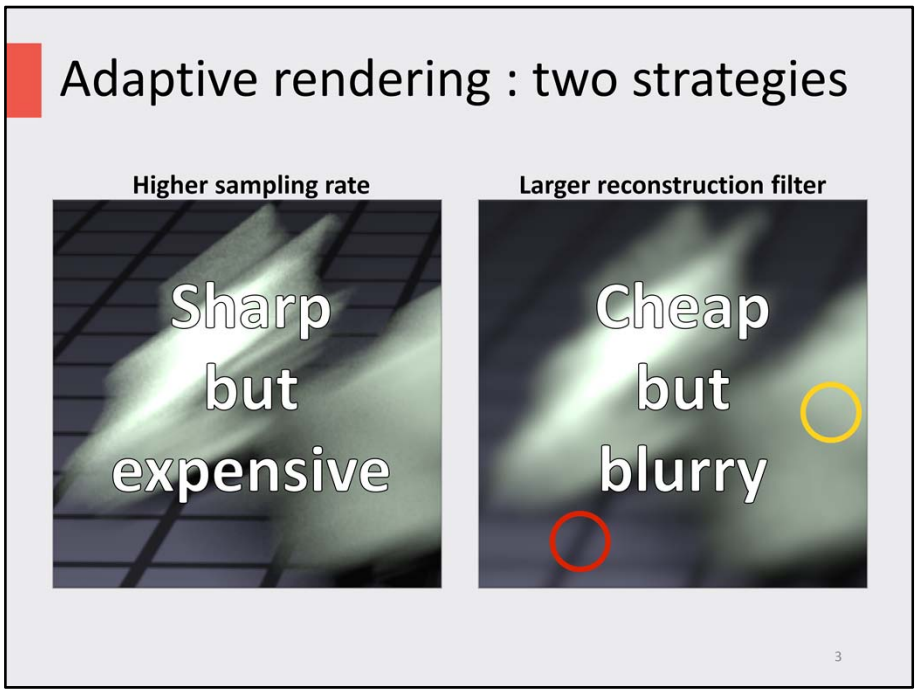
When doing image space adaptive rendering, we have two core strategies that we can leverage to reduce the variance of Monte Carlo renderings which I'll demonstrate with this simple scene.

The first strategy, shown on the left, is to simply increase the sampling rate. The second strategy is to use a larger reconstruction filter.

In practice, using a higher sampling rate gives sharp results but is computationally expensive. Using a larger reconstruction filter on the other end is cheap, but produces blurry results.

For instance, the lines on the floor, in the red circle, are now completely blurred out, which is clearly not acceptable. However, on the animal body, in the yellow circle, the reconstruction looks fine, since the region is actually very smooth.

The goal of adaptive rendering will be to combine these two strategies in a way that they complement each other's strength: using larger reconstruction filters whenever possible, and otherwise reverting to higher sampling rates.



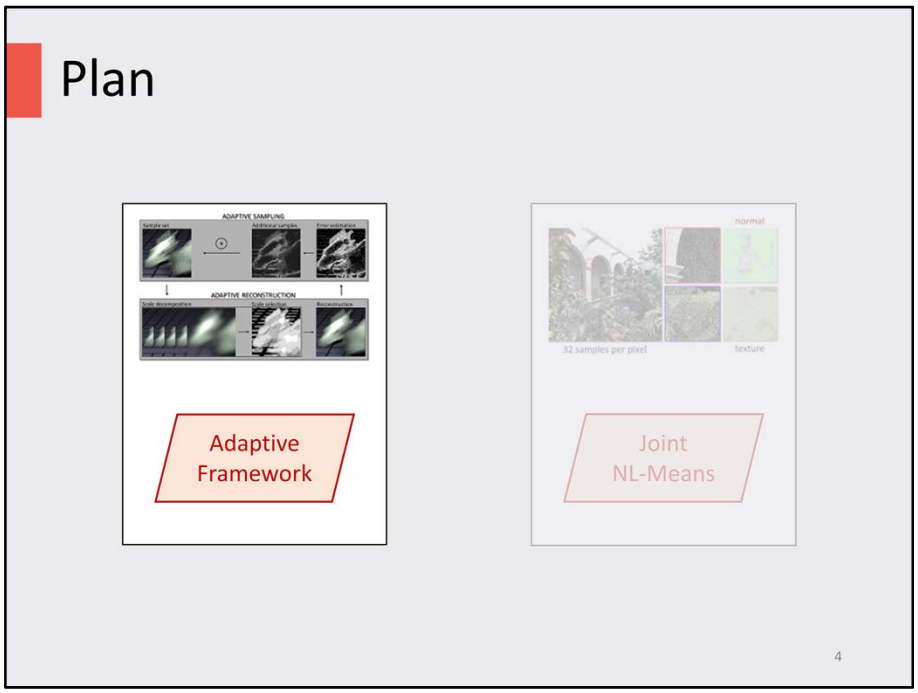
When doing image space adaptive rendering, we have two core strategies that we can leverage to reduce the variance of Monte Carlo renderings which I'll demonstrate with this simple scene.

The first strategy, shown on the left, is to simply increase the sampling rate. The second strategy is to use a larger reconstruction filter.

In practice, using a higher sampling rate gives sharp results but is computationally expensive. Using a larger reconstruction filter on the other end is cheap, but produces blurry results.

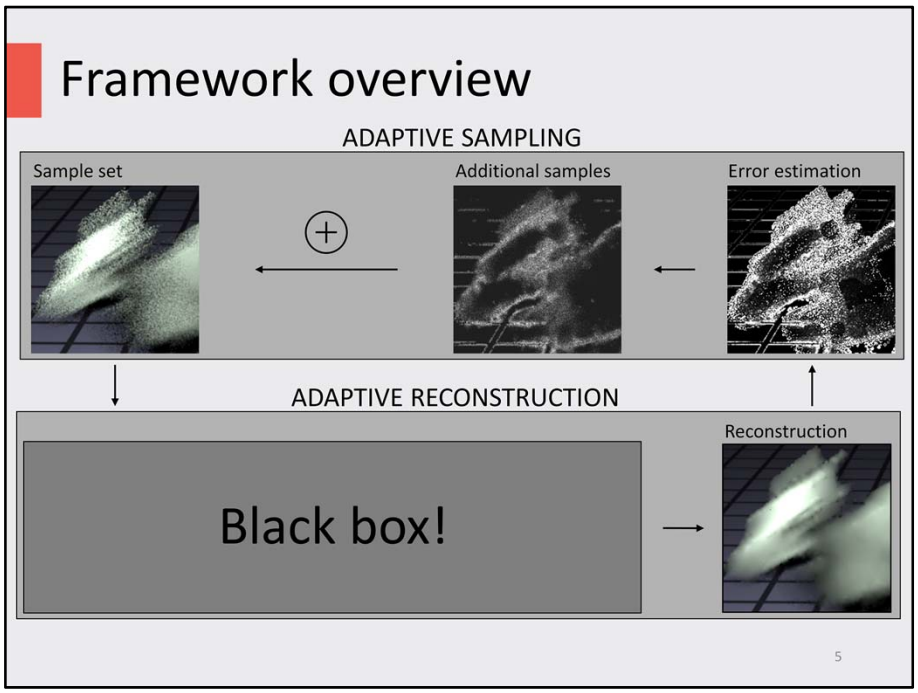
For instance, the lines on the floor, in the red circle, are now completely blurred out, which is clearly not acceptable. However, on the animal body, in the yellow circle, the reconstruction looks fine, since the region is actually very smooth.

The goal of adaptive rendering will be to combine these two strategies in a way that they complement each other's strength: using larger reconstruction filters whenever possible, and otherwise reverting to higher sampling rates.



Here is the plan of the presentation. I will first present the core adaptive framework which is at the heart of most recent adaptive rendering methods.

I will then cover in more detail one possible implementation of this framework based on the joint NL-Means filter. Starting from the original NL-Means filter, we will see how we can gradually improve its performance through multiscale filtering and leveraging more information from the rendering process.

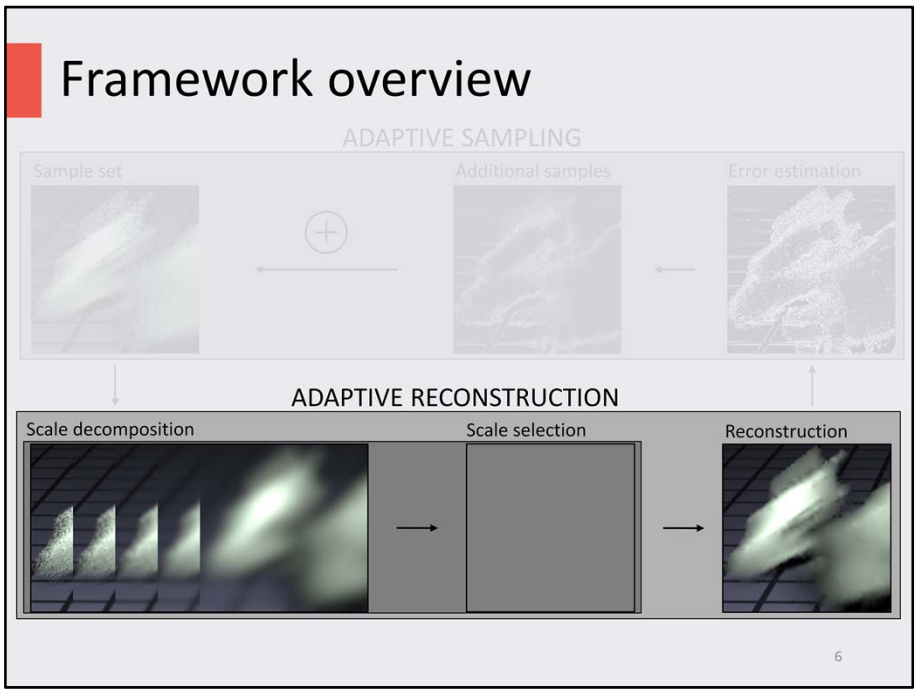


Here are the two core steps of the framework: the adaptive sampling and the adaptive reconstruction.

We start with an initial noisy sample set.
We then feed this sample set into a black box, out of which will come a reconstruction.
We will then estimate the residual error in this reconstruction, which we'll use to drive the next sampling pass. This residual error corresponds to the residual variance left after filtering and the bias introduced by the filtering.
We will not discuss the problem of adaptive sampling today, for the purpose of this presentation it is enough to know that we distribute more samples in regions with larger residual errors. Actually, the amount of samples taken is proportional to the expected reduction in error that these new samples would bring.
Finally, these new samples are added to the initial set.

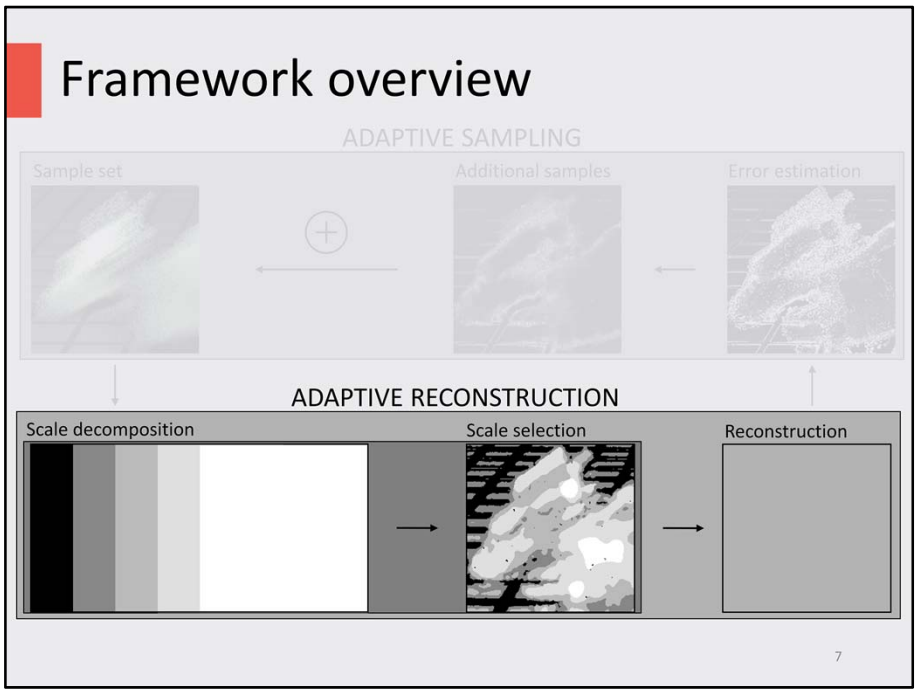
We can then keep on iterating over these steps until we run out of our time or sample budget. We could alternatively use an error threshold to detect convergence.

The main freedom we have in this framework is what we put in this black box. But for now, we'll start with a simple filterbank of isotropic Gaussian filters.

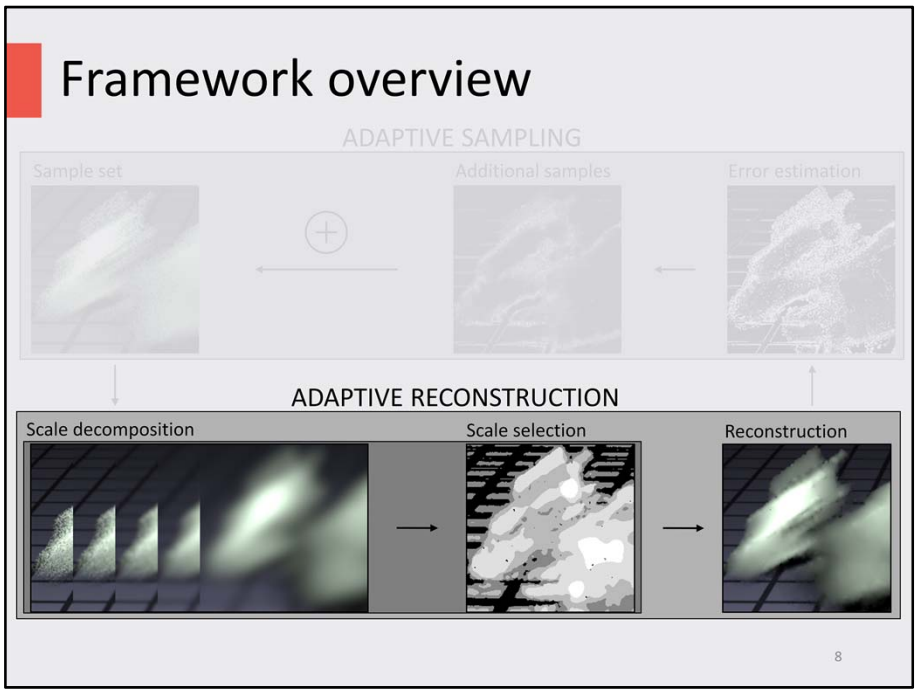


Our filterbank consists of five isotropic Gaussian filter with increasing bandwidths.

Our first goal will now be to figure out which of these five scales gives us the best reconstruction, on a per-pixel basis.



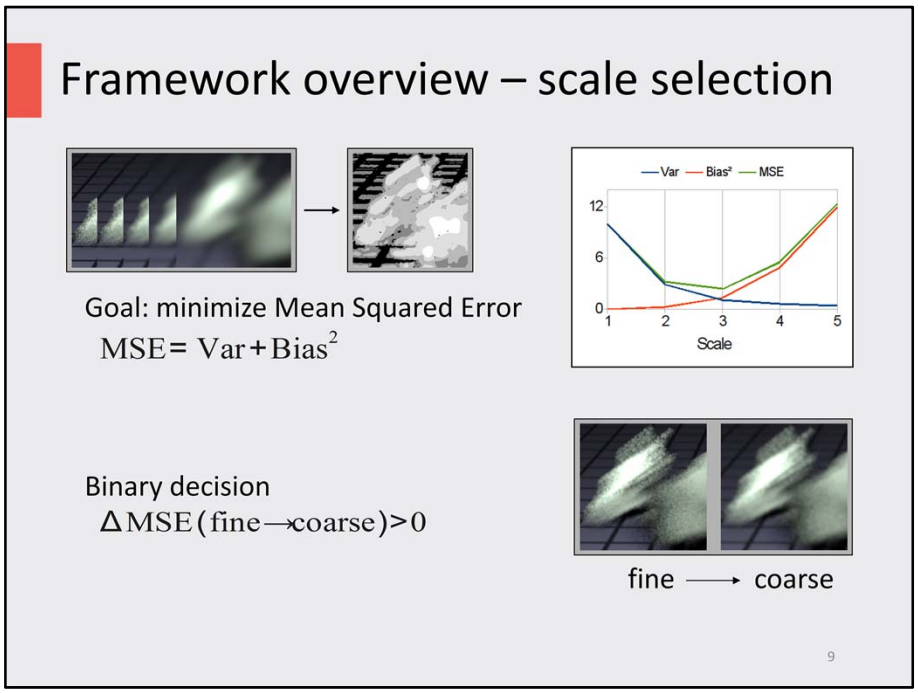
Using the following color scheme, this is what the scale selection looks like. We will see in a minute how this selection was performed.



We now take the first scale, and we take the pixels where it was selected, that is, the pixel that are black in the scale selection, and move them to the reconstruction.

Again for the second scale, we move all pixels where this scale was selected to the reconstruction.

We then do the same for the third, fourth and fifth scales.



At this point, the main question left is how to perform the scale selection.

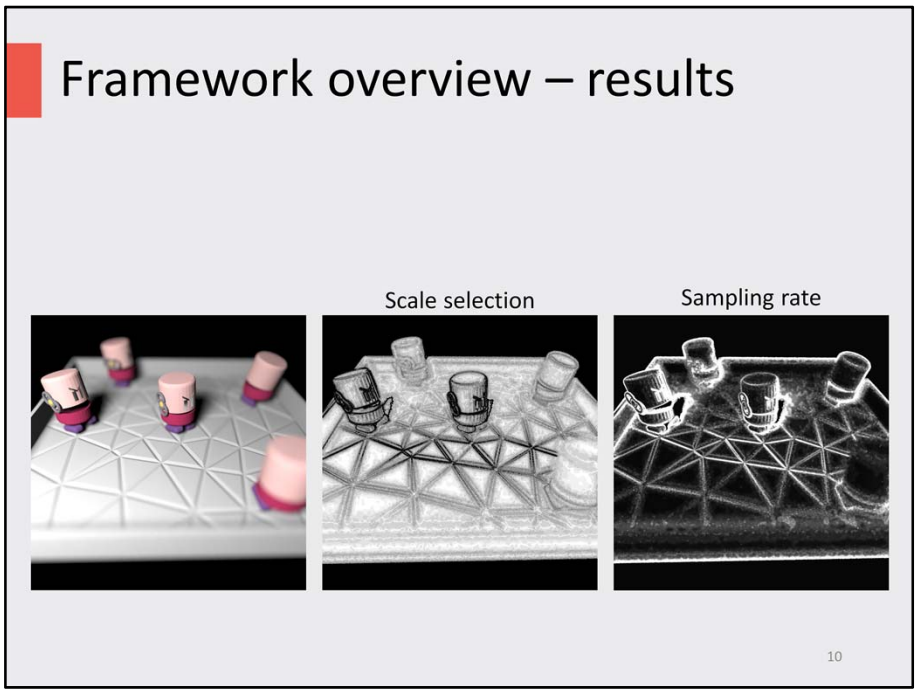
Our goal in this selection will be to minimize the mean squared error, which is the sum of the filtered output variance and squared bias.

As you see in this plot, the variance decreases monotonously as we move from the finer scales to the coarser scales.

The squared bias however increases as we go to coarser scales and the MSE, which is the sum of the two, is minimized in this case at the third scale.

Given this observation, the scale selection process is now straightforward, we simply select the scale at which the MSE starts increasing.

Naturally, the challenge will be in estimating the MSE, and we will come back to this question later on, but for now let us just assume that we have access to this information.



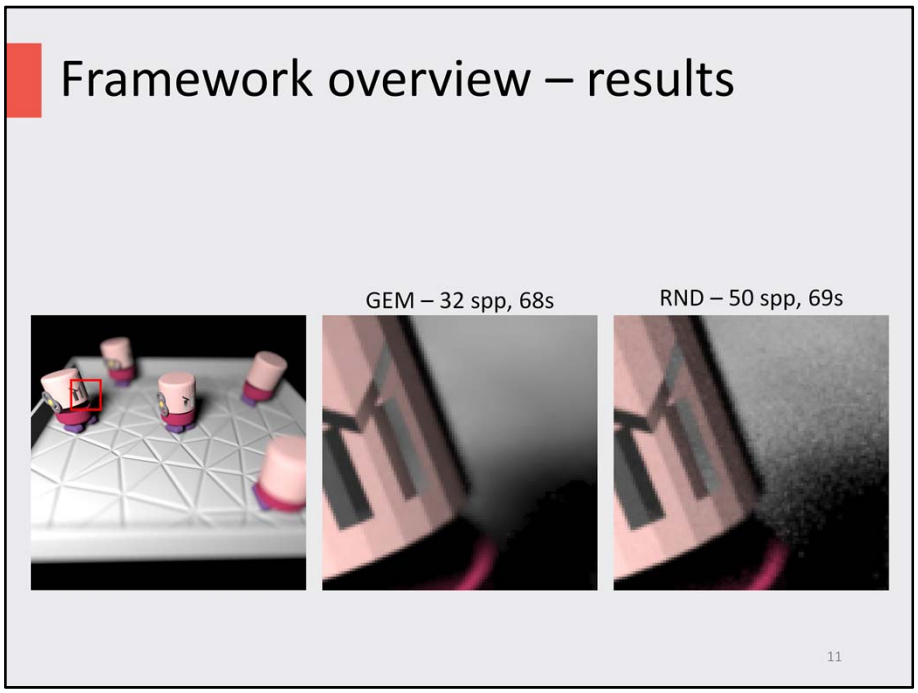
Here is now a result that can be obtained using this adaptive framework, using a filterbank of isotropic Gaussian filters.

This is a scene that features depth of field and area lighting.

In the middle, we show the scale selection, where black corresponds to the finest scale, that is, the smallest filter bandwidth, and white corresponds to the largest filter bandwidth. You'll notice that we use larger bandwidths in regions that are out of focus or locally smooth, while we use small bandwidths along edges in the regions in focus.

The sampling map on the right, essentially mirrors the scale selection: we are putting more samples along edges in focus, while smooth regions and out of focus regions use a much lower sampling rate.

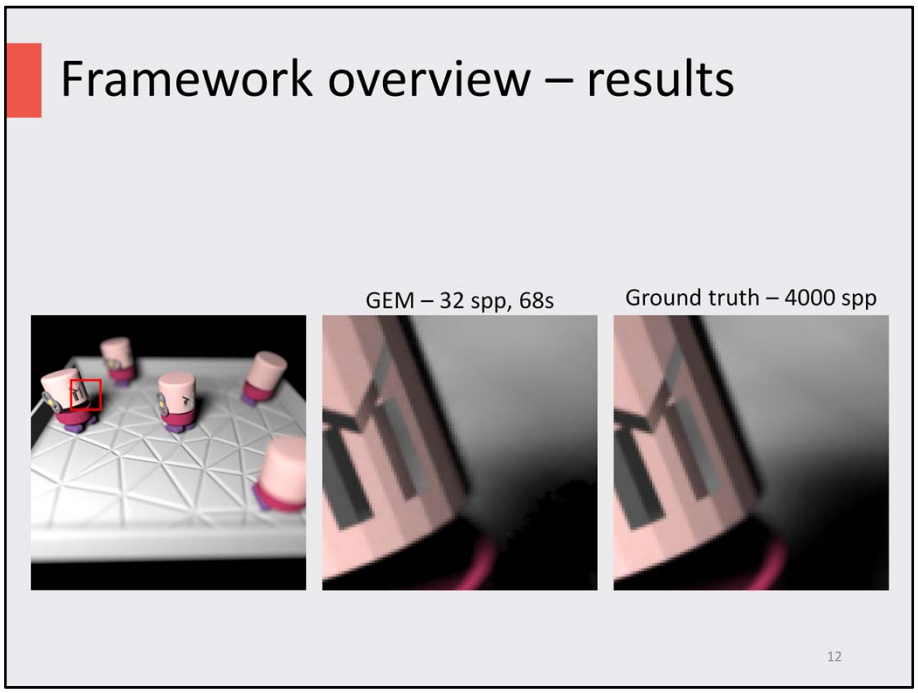
In practice this result is closely related to adaptive rendering methods based on frequency analysis, where the goal was to figure out the optimal per-pixel filter bandwidth and sampling rate according to the sampling theorem, except that we get to this result by minimizing the MSE.



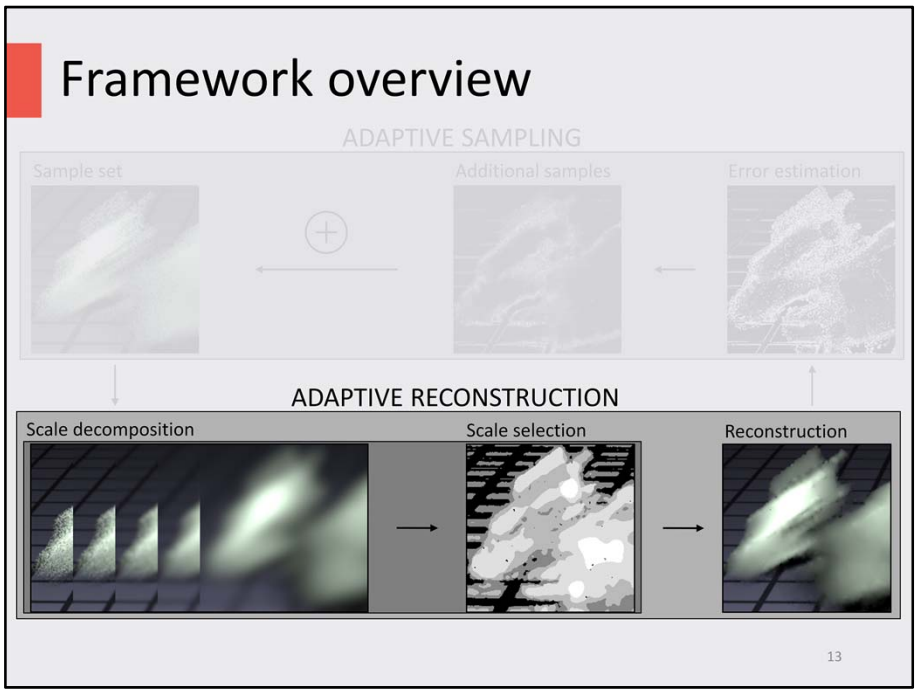
Here is a close-up view of a character head in focus with some hard edges, while the ground shows a smooth shadow due to area lighting.

On the right, we have an equal rendering time using standard Monte Carlo rendering with random samples.

Even though we use fewer samples, we still get a significantly better image in the end.

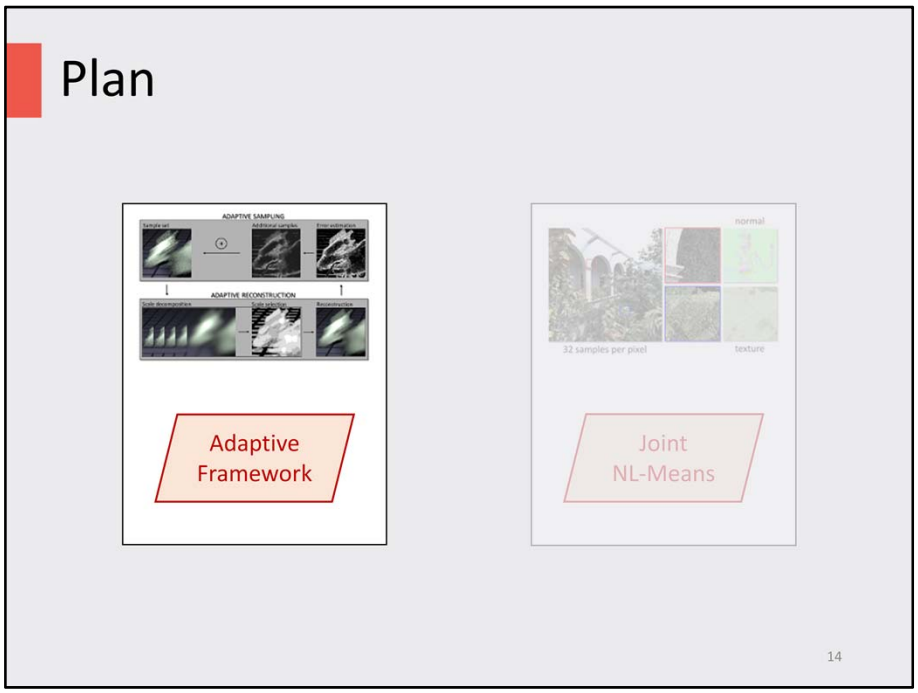


And here is a comparison with a ground truth rendering using 4000 samples per pixel.



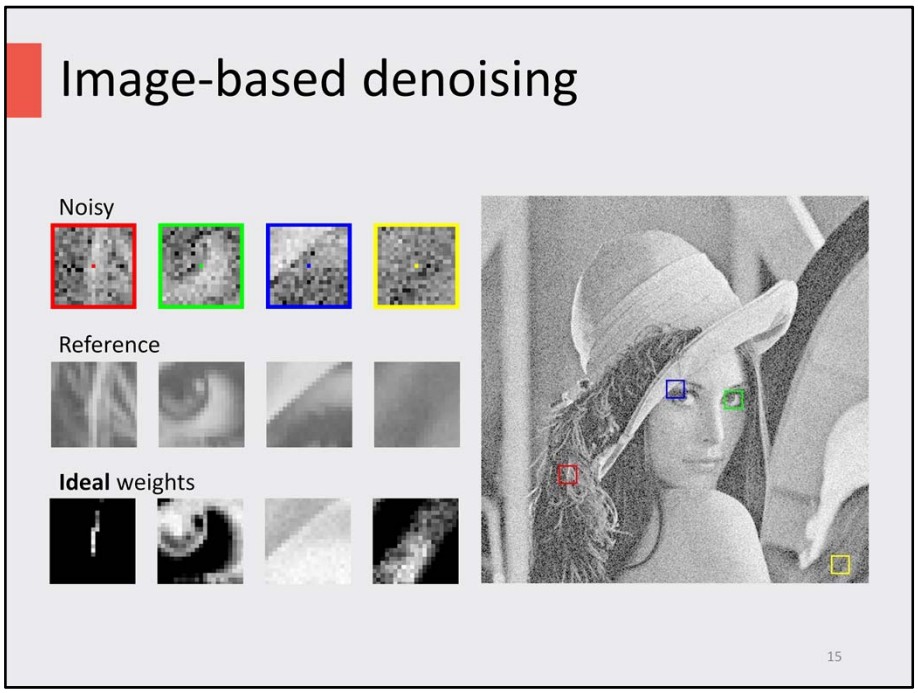
Going back to our framework, let's now see what we could do to improve the reconstruction itself.

This is actually the main advantage of image-space a posteriori methods: we have a wide range of powerful filters that we can leverage in order to get a better reconstruction using the same input sample set.



I will now describe such an implementation using the joint NL-Means filter.

Again, this implementation leverages concepts that are common to many other implementations.



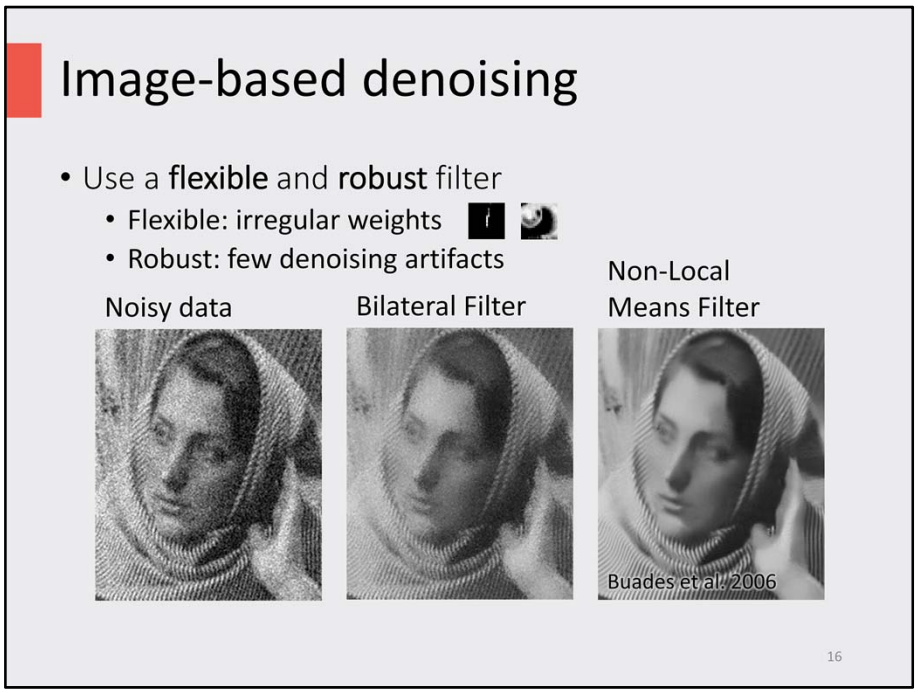
Let us go back for a moment to the core problem of image-based denoising.

We have here the lena image to which we added Gaussian noise.

Our goal will be to reconstruct each pixel as a weighted average of its neighborhood.

Here are the neighborhood before we added the noise, and the corresponding ideal filter weights. These weights were computed by generating 10 thousand noisy versions of the lena image and computing the weights that gave the overall best reconstruction on this whole set.

There are two things to notice about these weights. First, they differ drastically from pixel to pixel. Second, the shape of the weights often cannot be approximated using simple filters like a Gaussian filter, even an anisotropic one.



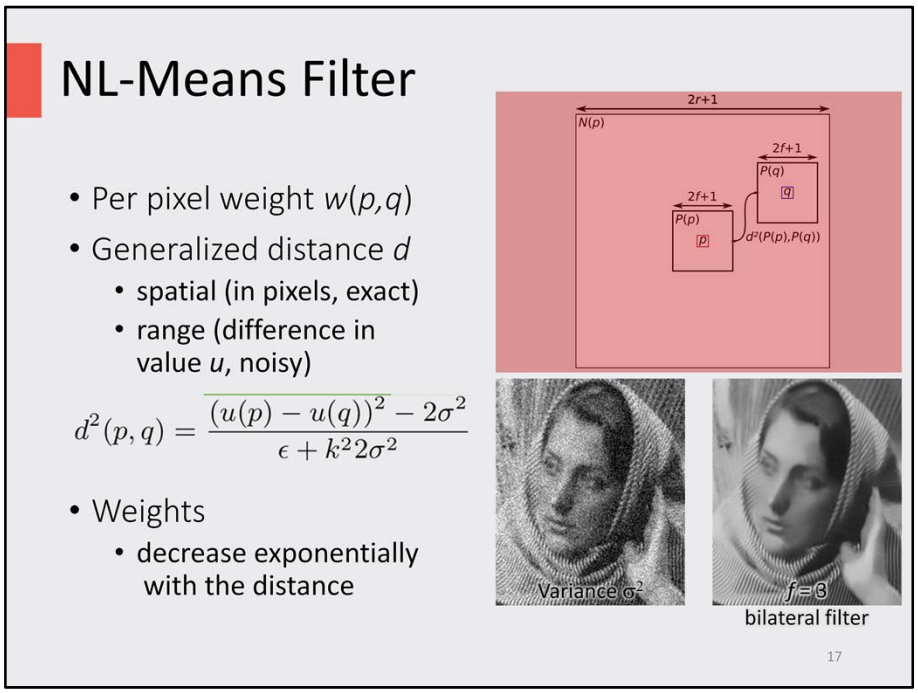
In practice, we do not have access to perfect information, but we would still like to use a filter that can approximate them as well as possible.

For this, we'd like to use a filter that is both flexible and robust.

A "flexible" filter is a filter that can take irregular weights such as those shown on the previous slide.
A "robust" filter is a filter that has as few denoising artifacts as possible.

One filter that fits the flexibility requirement is the bilateral filter, unfortunately it is not really robust to noise as we can see here. Even though we managed to remove some of the noise, the output is still fairly grainy.

Instead, we will make use of the non-local means filter, initially proposed by Buades et al. in 2006, which is a generalization of the bilateral filter that greatly improves its robustness to noise.



Let us now see how the NL-Means filter operates.

The goal of the filtering process is to compute the reconstructed value of a pixel *p* as a weighted of its neighborhood. We have here the pixel *p* and one its neighbors *q*.

We now want to compute the weight of *w*(*p*,*q*) of the contribution of *q* to *p*. The weight will be based on the affinity between these two pixels, as measured using a generalized distance *d*. This distance has two components:
- A spatial component, measured in pixels. In this case we simply restrict ourselves to a square neighborhood of side 2*r*+1.
- A range component, that measures the difference between the two pixel values. This difference is actually noisy, since our input is noisy.

In practice, because of the noise in our input, the squared difference between the two pixel values will be positively biased. We can however cancel out this bias by subtracting the variance of the squared difference. Note that we assume that the noise in the two pixels is uncorrelated.

We then normalize the squared difference by dividing it by the same variance, which we multiply by a user parameter *k*, which allows the user to control the aggressiveness of the

filter. A larger *k* value gives a more aggressive filter.

Finally we add a small epsilon to prevent divisions by zero.

Now that we have the distance between the two pixel values, we can compute the weight using an exponential function that ensures that the weight rapidly goes down to zero as the distance increases.

Up to this point, this description also corresponds to the behavior of the bilateral filter, but the NL-Means filter now adds a key step: distances will be computed (and then averaged) over square patches of side $2f+1$.

Setting $f=0$ corresponds to the behavior of the bilateral filter, which gives the grainy output that we have already seen previously. Setting $f=3$ gives this much improved result. The improvement comes from the fact that our patches consist of 7x7 pixels, which gives us 49 distinct distance evaluations that are averaged together. The averaging greatly reduces the distance estimation variance, yielding more robust weights and therefore a more pleasing output.

NL-Means

128 samples per pixel

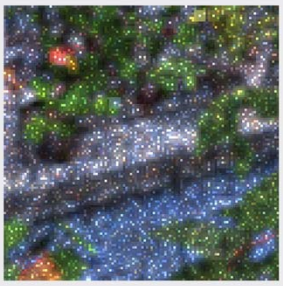
18

Now that we know how this NL-Means filter operates, let us see how we can apply it to our Monte Carlo renderings.

NL-Means – Non-uniform variance

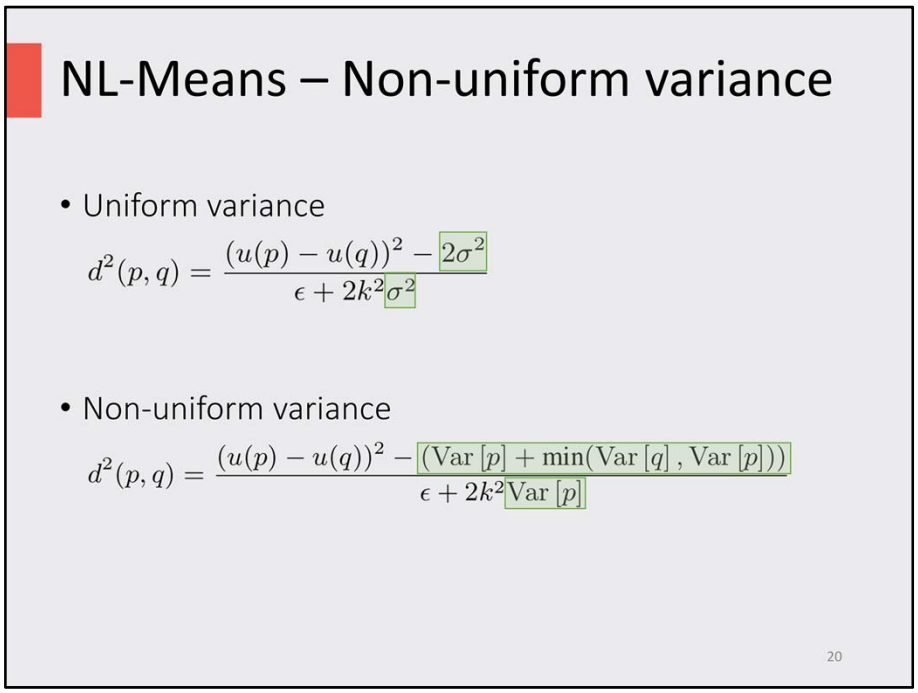
128 samples per pixel

Variance



19

The first issue we will face is that the variance of Monte Carlo renderings is not uniform across all pixels, but the NL-Means filter formulation assumes uniform variance.



Here is the squared distance computation used in the NL-Means filter.

We will start by replacing the 2sigma^2 at the numerator by the sum of the variance at pixels *p* and *q*. In practice though, for pixel *q*, we will use the minimum between the variance at *p* and *q*. This is useful to prevent noisy bright regions from growing into darker ones.

At the denominator, we will simply replace the variance sigma^2 by the variance at *p*.

NL-Means – Variance estimation

- Independent samples: sample mean variance

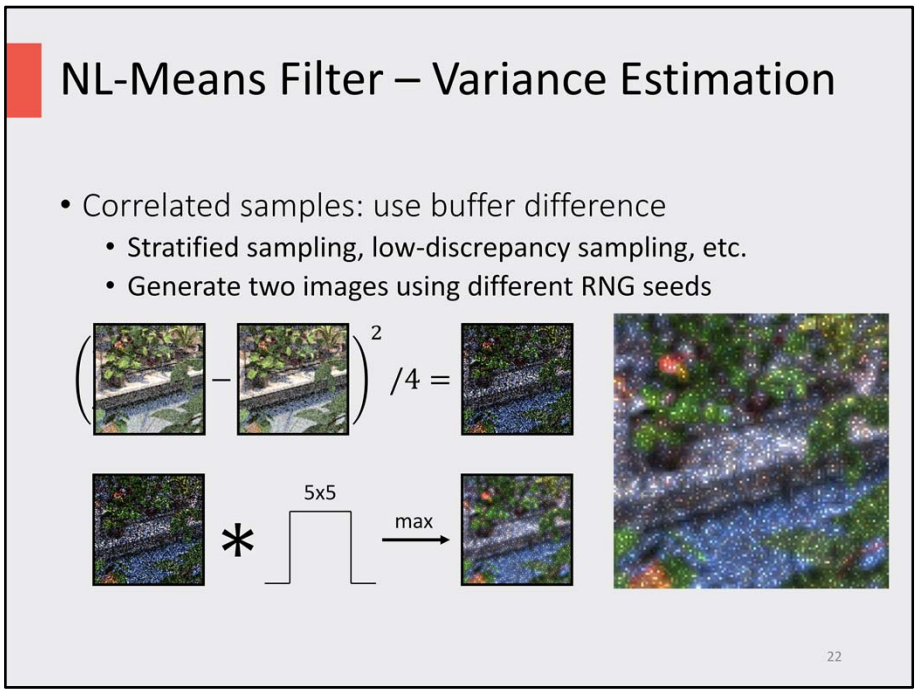
$$\mathrm{Var}[p] = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right) / n$$

21

The new step will be to estimate the variance of the pixel means in our rendering.

When using random samples this can be trivially done by directly computing the sample mean variance, if we assume the variance of all samples falling within a pixel to be constant.

This gives the result shown on the right.



In practice however we usually use correlated samples in Monte Carlo rendering in order to reduce the variance of the integrator. We could be using stratified sampling, low-discrepancy sampling or even Metropolis Light Transport.

With correlated samples, the sample mean variance formula is not applicable, and we instead propose to use a simple approach which consists of rendering two images using different random number generator seeds.

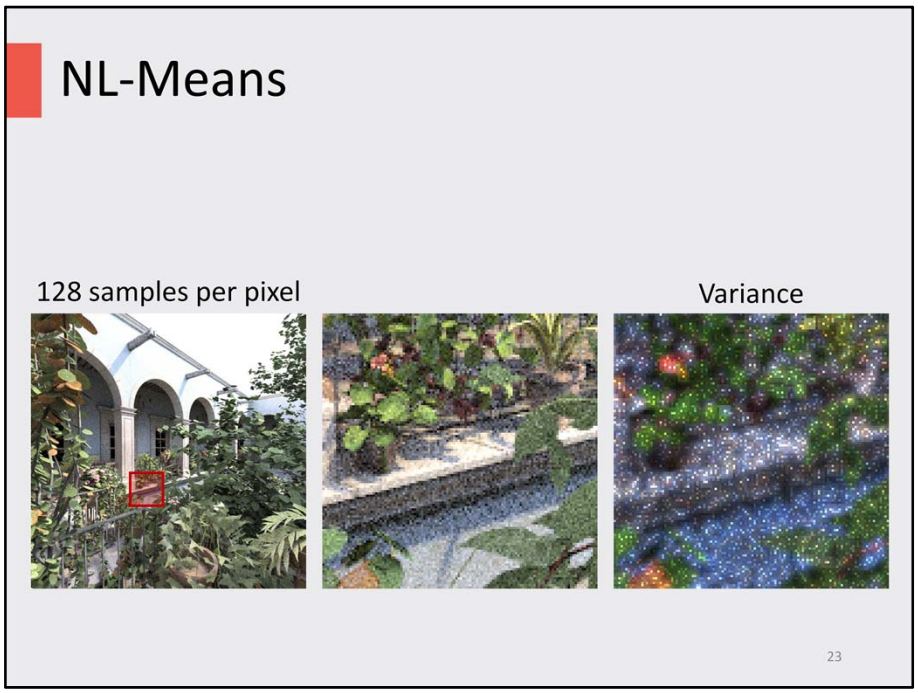
Even though we could not separately compute the variance of these two images, since each was obtained using correlated samples, we can compute the variance of their mean, since using different RNG seeds ensures that the noise is uncorrelated between the two images.

We compute the variance of the mean by taking the squared difference between the two images divided by 4. This yields an extremely noisy (but unbiased) variance estimate, which is unsurprising given that it was computed from a 2-sample population: the two independent images.

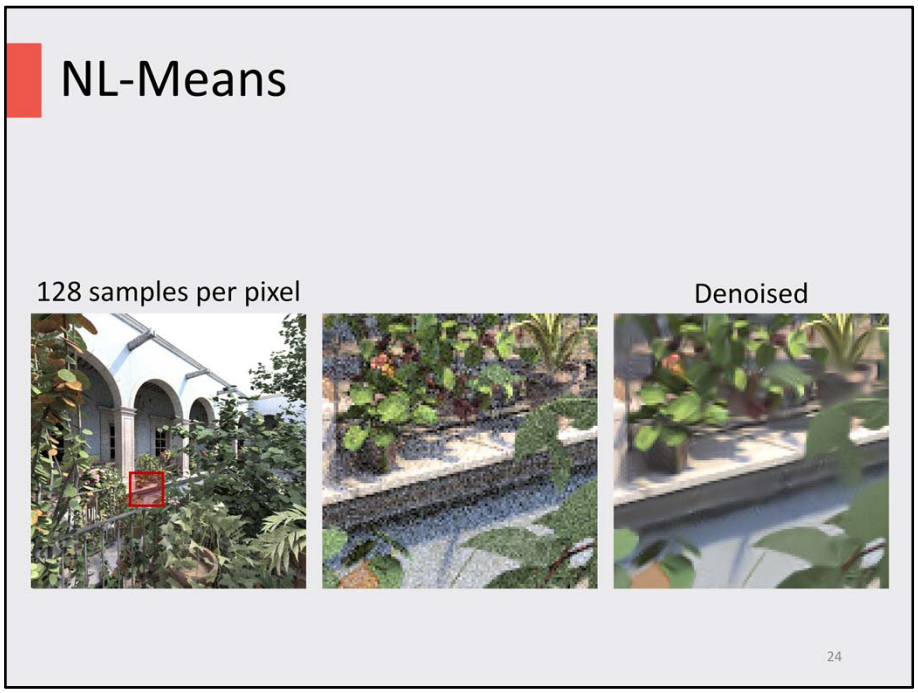
The variance of our variance estimate is problematic since we will significantly under-estimate the variance for many pixels. All pixels with a dark value in the variance map

correspond to pixels where the variance is estimated to be very low. This will effectively prevent the filter from properly removing the noise.

We can solve this issue by simply applying a 5x5 box filter to our noisy variance estimate, and taking the maximum between the box filtered variance and the initial variance estimate. Taking the maximum of these two is useful because we want to preserve positive outliers in the variance estimation, since these corresponds to outliers in the image buffer which have very different values from their neighbors, and the high variance estimate ensures that these outliers will be filtered out.



Given the new formulation of the NL-Means distance computation and our variance map, we are now ready to apply this filter to our Monte Carlo rendering.



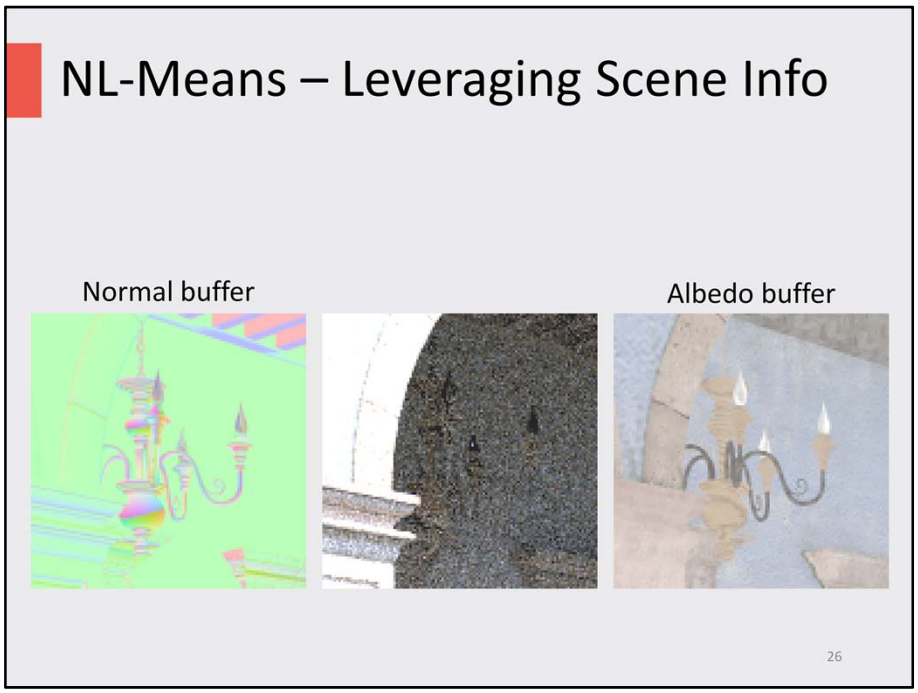
This is the result we obtain on this crop. Overall, we get a nice smooth output, while preserving the strong edges of the scene.

Let us now look at a different crop of the same scene.

In this crop, we see a chandelier hanging from the ceiling. This part of the scene is lit by indirect illumination and it has a high level of noise and a relatively low contrast.

Using the NL-Means filter on this region again yields a smooth output, but in practice we blurred out all the geometric details of the scene, and the final result is not satisfying.

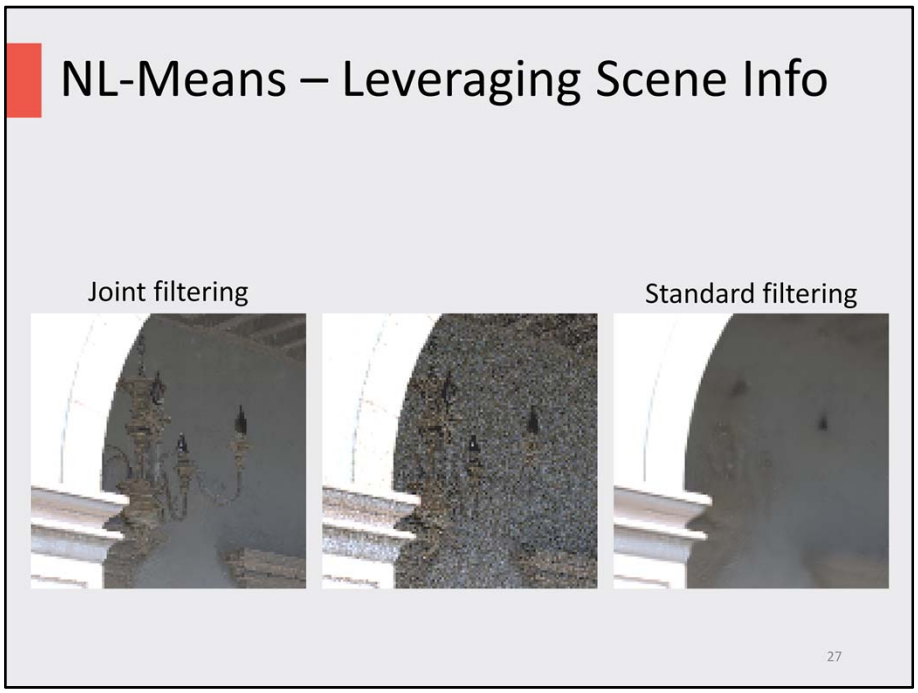
This illustrates a fundamental limitation of image-space denoising techniques leveraging the pixel color: if the signal is drowned in noise, it cannot be properly reconstructed, since there is no way of detecting what are actual edges to be preserved and what is noise.



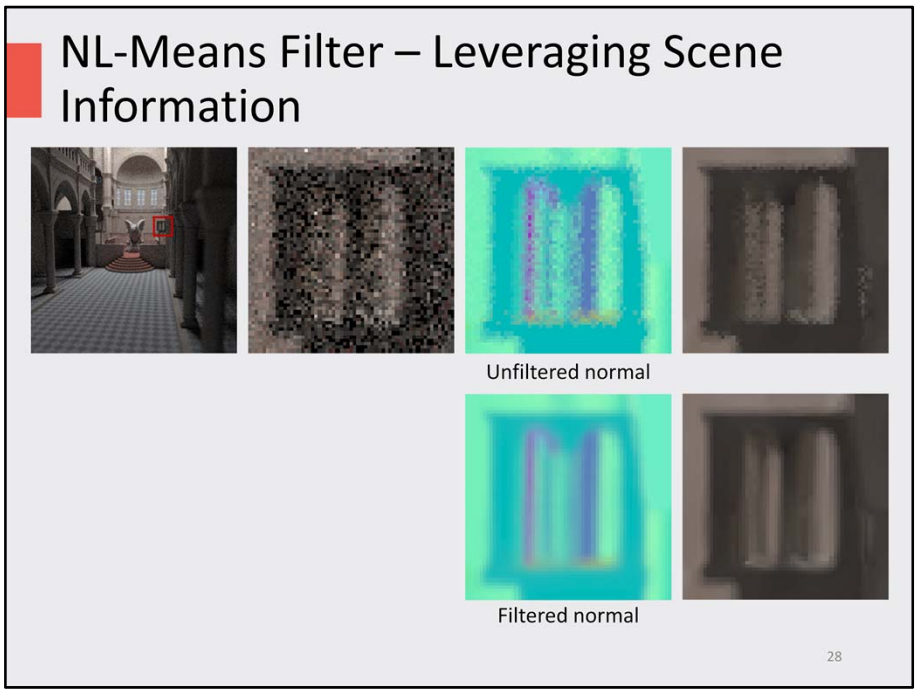
Fortunately, in the rendering context we have access to a lot of information about the scene beyond the pixel color.

For instance, we can extract the normal and albedo buffers, shown here. These are obtained as a by product of the rendering process and nicely capture both the geometric details of the scene, as well as the basic appearance of the materials. Furthermore, at the same sampling rate, these are virtually noise free.

We can now leverage these feature buffers in a joint filtering scheme. Instead of simply measuring the difference between two pixel values according to the pixel color, we will also measure the difference between pixel normals and pixel albedos . We then use the most constraining of these measured differences to compute the filter weights in order to preserve the edges encoded in these buffers.



Here is the result, on the left, of this joint filtering scheme, where we see that we significantly improved the reconstruction of the fine details of the scene, despite using the same set of samples as input.

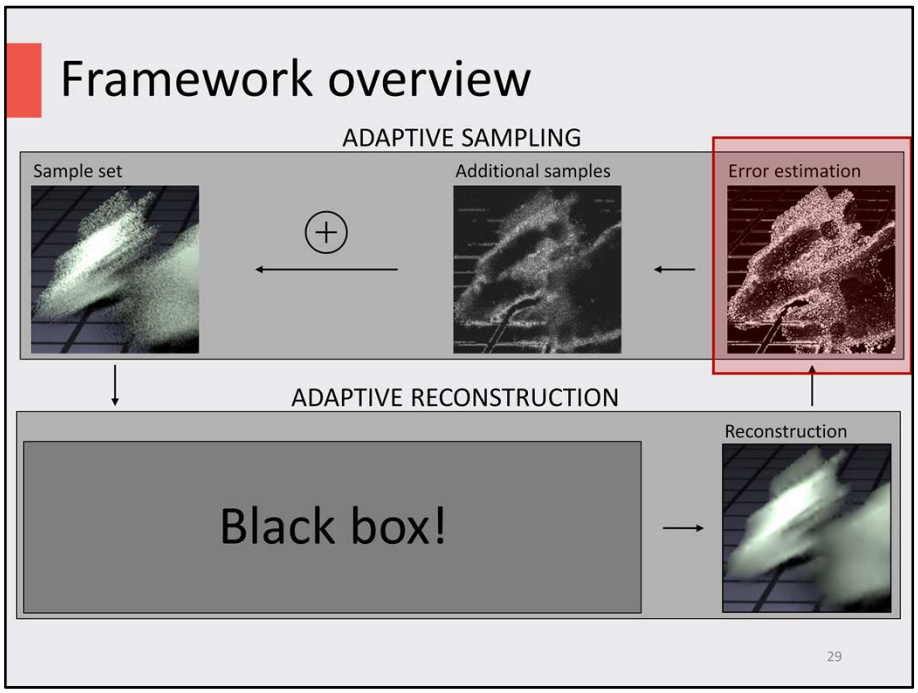


The joint filtering example we just saw assumed that the auxiliary feature buffers were noise free, which is not necessarily the case. These feature buffers could have variance because of aliasing within a pixel, depth-of-field (as is the case in this example), or motion blur.

If we directly make use of the noisy normal buffer to guide the filtering of the pixel color, we get the result on the right. While we did remove most of the noise, the noise structure of the normal buffer has be transferred to the output.

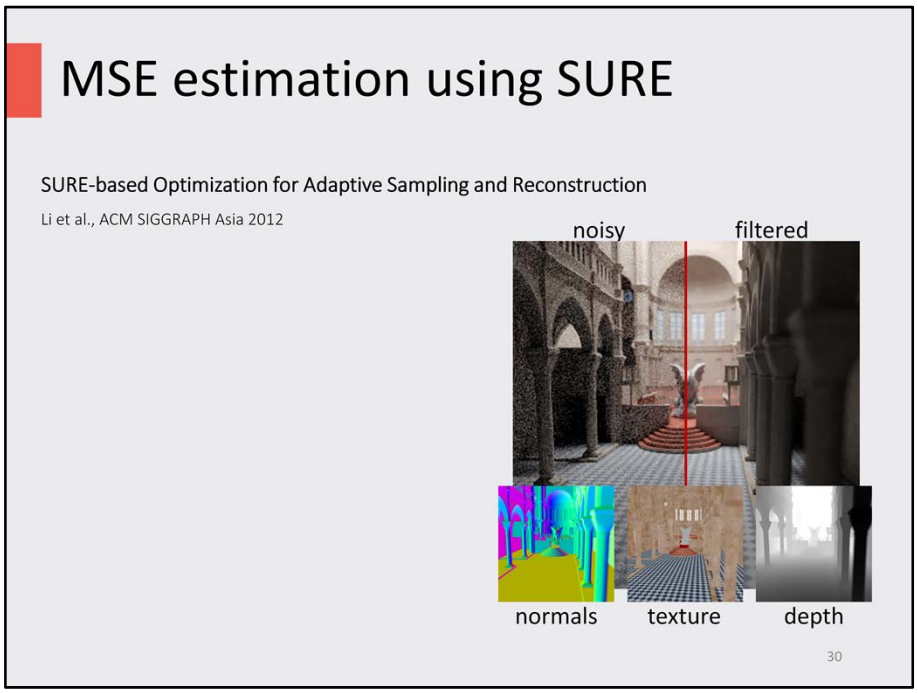
We can solve this issue by simply prefiltering the normal buffer, and then using this filtering normal buffer to guide the denoising of the pixel color, which gives a smoother output as seen on the bottom-right result.

This approach can be seen as a way of breaking down the overall denoising problem into simpler problems. The normal buffer in this case only suffers from noise due to the depth-of-field, and the signal to noise ratio is still fairly good and we can get a good reconstruction of the normal buffer. Having solved the simple problem of reconstructing the normal buffer then helps us solve the more challenging problem of reconstructing the pixel color.



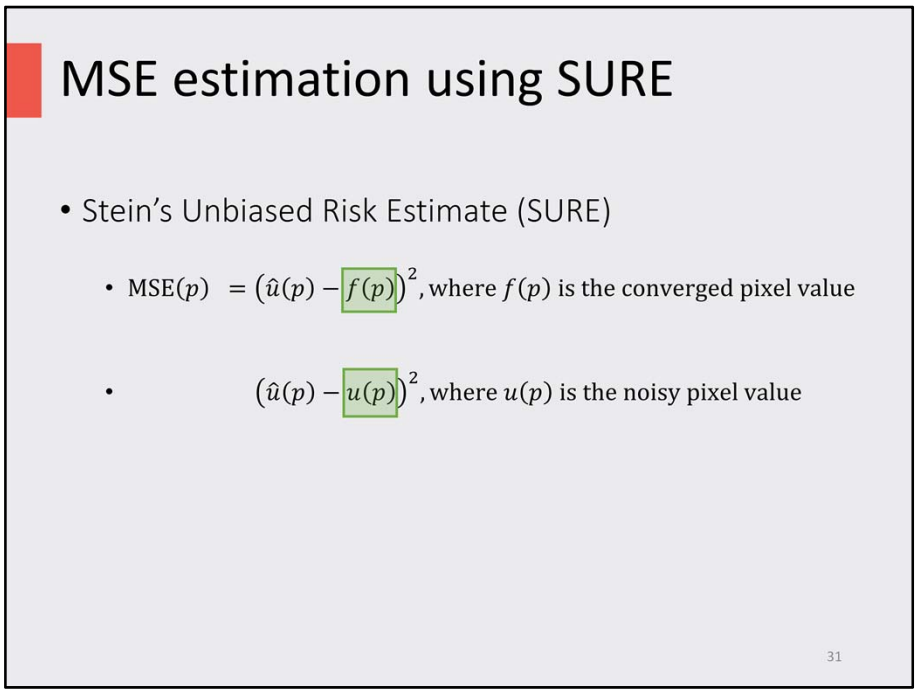
Let us now go back to our framework overview.

Having filtered our noisy input set, we now need to estimate the residual error of the reconstruction in order to drive the adaptive sampling. In practice, we will use the Mean Squared Error (MSE) of the reconstruction as an error metric.



We will compute the error of the filter output using SURE, which is an unbiased estimator of the MSE.

The use of SURE in the context of denoising Monte Carlo rendering was first proposed by Li and colleagues in a recent SIGGRAPH Asia paper, though SURE has been used previously in other fields to optimize filter parameters.

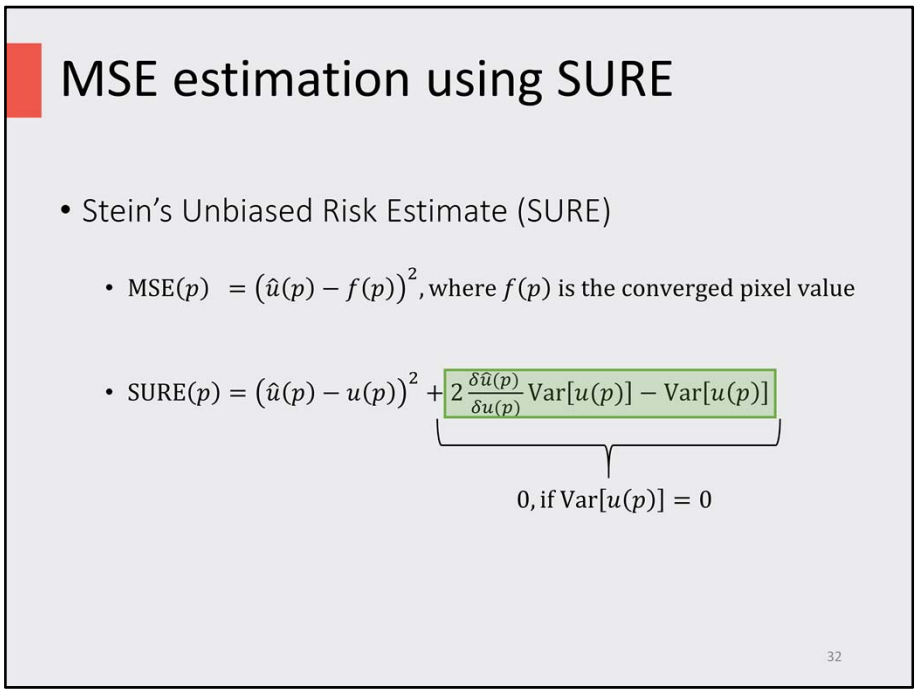


SURE stands for Stein's Unbiased Risk Estimate. In order to introduce SURE, let us first look at the definition of the MSE.

The MSE is given by the squared difference between the filter output $\hat{u}(p)$ and the converged pixel value $f(p)$. Of course, in practice we do not have access to the converged pixel value since it is precisely what we are trying to compute.

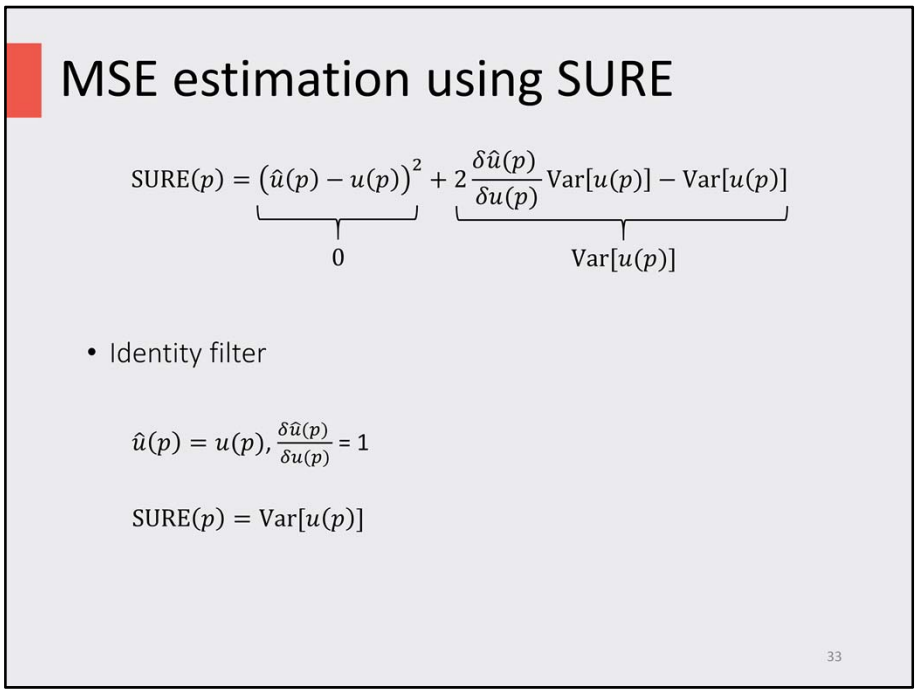
We however have access to the input noisy pixel value, so we can replace $f(p)$ with $u(p)$. In itself this is not really helpful though. Consider the case of the identity filter where $\hat{u}(p) = u(p)$. In this case, the measured error $(\hat{u}(p)-u(p))^2$ would be 0, suggesting that we have a perfect filter, whereas we simply have a filter that does nothing.

SURE therefore adds a second term, using the derivative of the filter with respect to its input, that addresses this problem.



SURE therefore adds a second term, using the derivative of the filter with respect to its input, that addresses this problem.

The first thing we can say about this second term is that it is equal to 0 whenever the variance of *u*(*p*) is 0, that is, if we give a converged image as a reference in SURE, we fall back on the MSE formula..

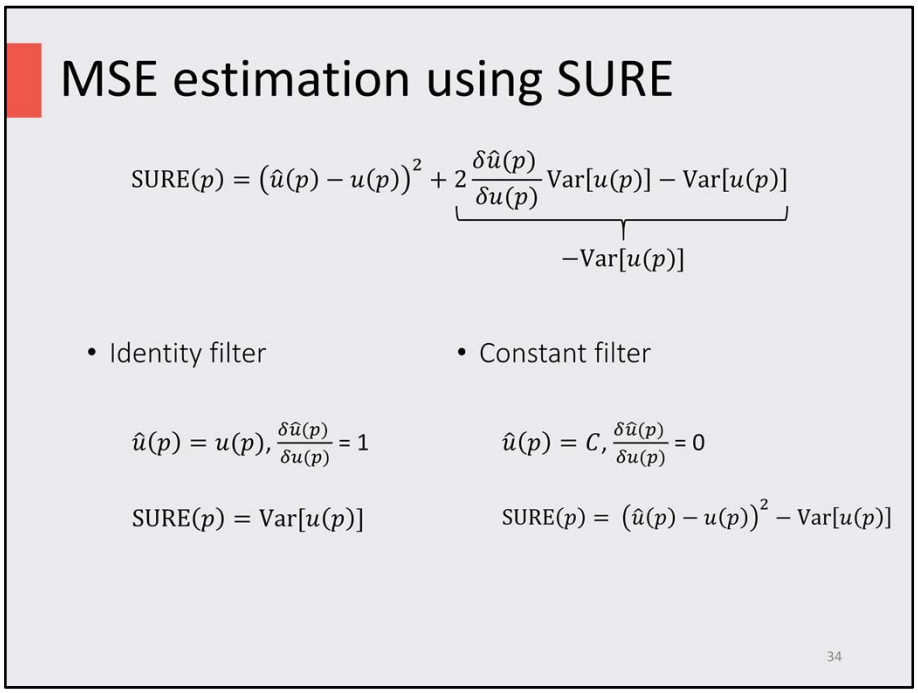


We will now simply give an intuition on the second term of the SURE equation, using two extreme cases.

The first case is the identity filter that we mentioned previously. With the identity filter, the derivative of the filter output with respect to its input is 1. Let us now see how this affects the two terms of SURE:

- The first term is now equal to zero.
- The right term is now equal to Var[$u(p)$].

The output of SURE for the identity filter is that the error of the output is the error of the input, that is, the variance of the input.



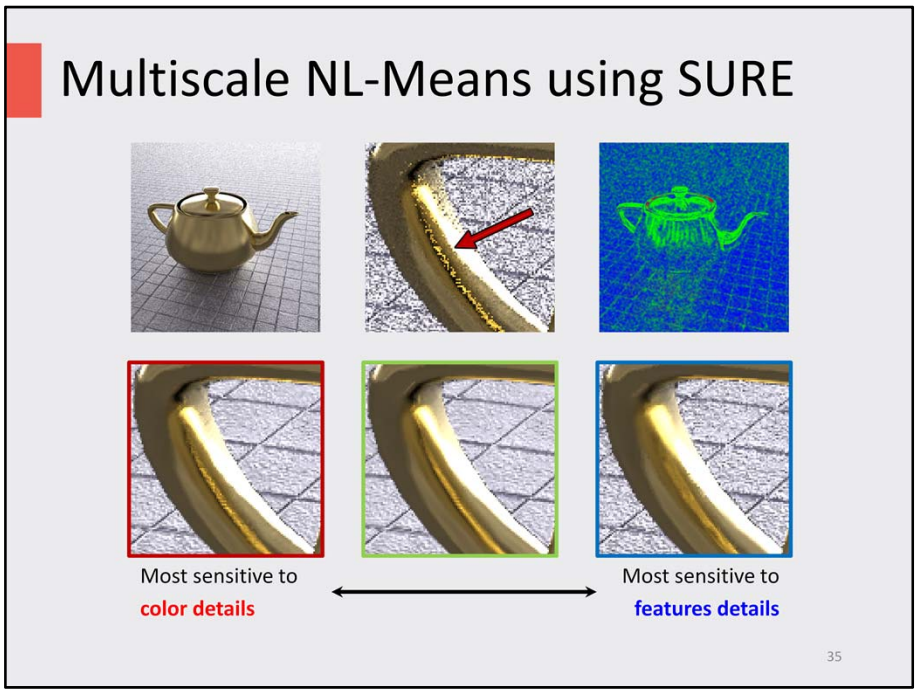
The second case is the constant filter where $\hat{u}(p) = C$, that is, the filter output is independent of the input. In this case, the derivative of the filter output with respect to its input is zero.

Let us now see how this affects SURE:
- The second term is now equal to –Var[$u(p)$].

The output of SURE for the constant filter is that the error of the output is the squared difference of between the output and the input minus the variance of the input. Again, this is the expected behavior, since the input has some variance that will positively bias the squared difference. We cancel out this bias by subtracting the variance.

In practice the derivative of the filter output with respect to its input can be seen as a way of interpolating between these two extreme cases.

In order to use SURE to estimate the MSE of our joint NL-Means filter, we need its derivative with respect to the input, which we will simply compute using finite difference.



Now that we have a way of estimation the MSE of the joint NL-Means filter output we can not only use it to drive the adaptive sampling stage, but also to perform scale selection in a multiscale filtering scheme.

We will illustrate this approach on this example of a glossy teapot on top of a bump mapped floor.

The bump map on the floor is difficult to distinguish from the noise, but are reliably encoded in the normal buffer and we would therefore want to give more importance to the feature buffers in this case.
On the other hand, the highlight on the teapot handle is due to a second bounce of indirect illumination and is not captured by any of our feature buffers, so we will have to rely on the pixel color to reconstruct it.

We propose to use three scale for this purpose. The first scale is more sensitive to color details, while the third one is more sensitive to feature details. The second scale offers a balance between these two. Notice how the glossy highlight is blurred out in the third scale, was the grainy texture on the ground is better preserved.

The scale selection, in the top right, shows the per-pixel weights of the three scales. The

red, green, and blue channels indicate the weights of the first, second, and third scales. As you can see, we primarily make use of the third scale on the ground, but make use of the second scale in regions with glossy highlights, since these are only captured by the pixel color. Also, we rarely make use of the first candidate filter, which is mostly useful in complex regions and regions that are nearly converged.

Multiscale NL-Means – results

Denoising – 16 spp, 19s

LDS – 33s, 19s

36

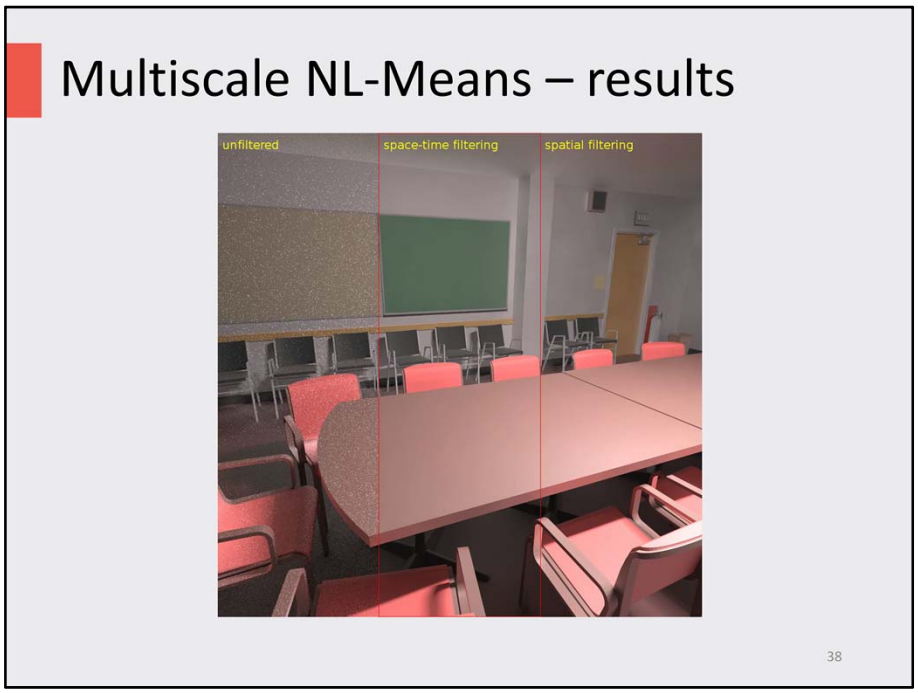
Here is the final result on the teapot scene, compared to an equal time rendering using standard Monte Carlo rendering with low-discrepancy sampling.

Again, despite using fewer samples, we obtain a result of significantly better quality.

Multiscale NL-Means – results

37

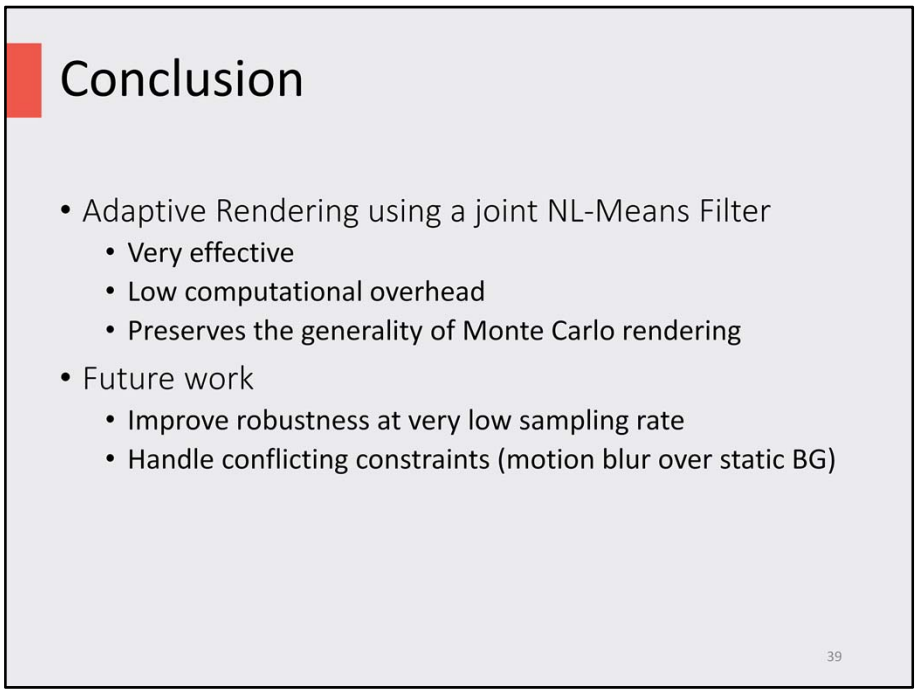
This is a comparison with a ground truth rendering.



This animation highlights the temporal coherence of our method.

On the left, we have the unfiltered noisy input, and on the right the result of our spatial filtering, where each pixel value is computed as a weighted average of its neighborhood. However, we have quite a bit of temporal flickering due to the residual low-frequency noise in out output.

We can trivially extend this approach to space-time filtering, where we filter not only spatially, but also across frames. This enforces some temporal coherence, since adjacent frames are reconstructed from overlapping sets of input samples, and significantly reduces the amount the flickering.



I'm now ready to conclude.

I've presented you with a set of three image-space adaptive rendering methods, the last of which offers state of the art results.

This class of methods have been surprisingly neglected for a long time, and most of the research has been done in the last five years. However, despite being conceptually very simple, these methods are also very effective. And this combination of simplicity and effectiveness is a great part of their appeal.

They also have the advantage of having a low computational overhead, but more importantly, they preserve the generality of MC path tracing. I think this is the most important point, since this generality is the main reason why people are interested in MC path tracing to begin with.

In terms of application, I think these methods will greatly facilitate the deployment of MC path tracing in production environments, and that they will be the key to enabling realtime and interactive MC path tracing.

Adaptive Rendering using Local Weighted Regression (LWR)

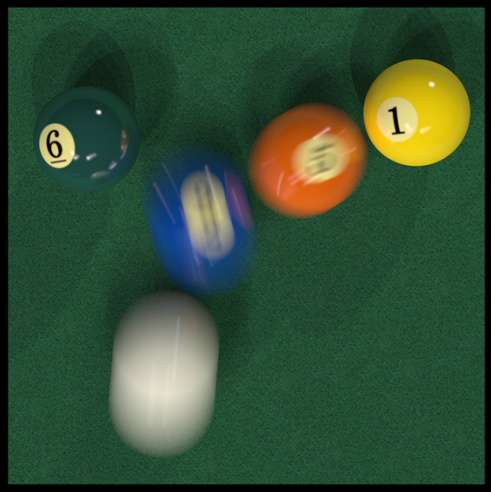
Sung-Eui Yoon
KAIST

Acknowledgements

- **Collaborators**
  - My students, and Bochang Moon; this is based on his thesis slides
- **Funding sources**
  - Korea Research Foundation
  - Ministry of Knowledge Economy
  - Samsung
  - Microsoft Research Asia
  - Adobe
  - Boeing

Monte Carlo Ray Tracing

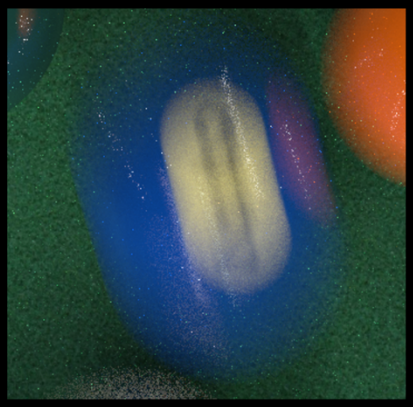
- **Produces photo-realistic rendering effects**



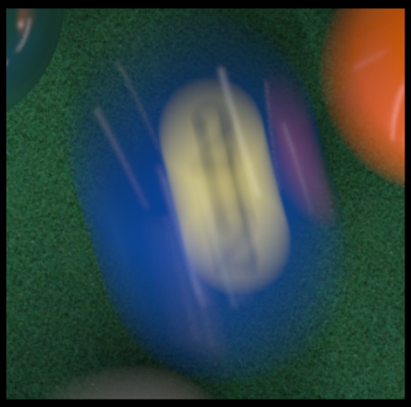
Motion blur effects

Monte Carlo Ray Tracing

- **A large number of samples should be traced.**

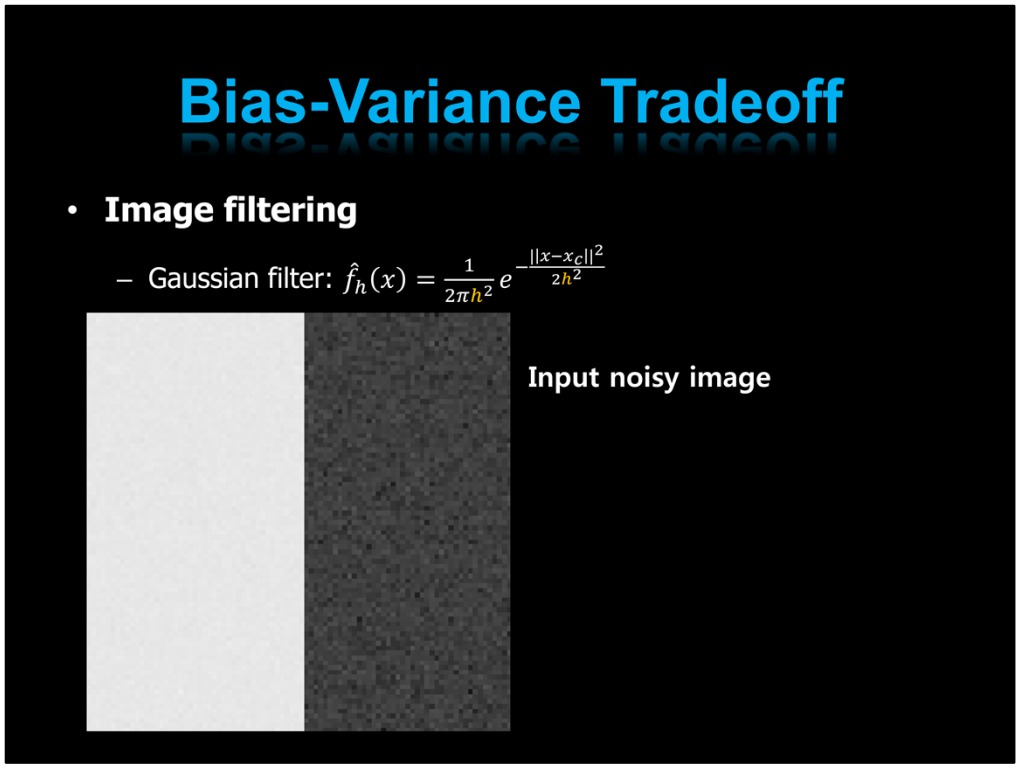


N = 53



N = 64 K

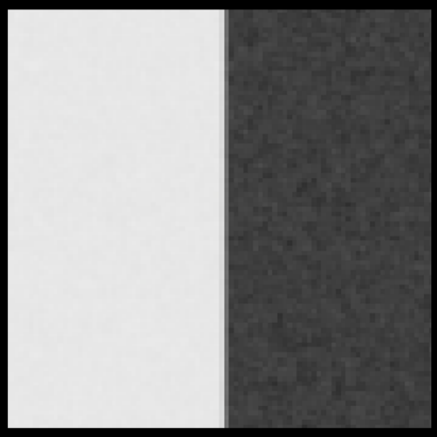
I will only briefly talk about the adaptive rendering.



In this work, we aim to compute visually pleasing results for MC generated images.
There can be different methods for the goal. In our case, we aim to design an image filtering tailed to MC rendering.
Suppose that we use a simple Gaussian filter on the image. It has a parameter, bandwidth parameter, h. Also, suppose that we have an image of black/white with random noise.

Bias-Variance Tradeoff

- **Image filtering**
  - Gaussian filter: $\hat{f}_h(x) = \frac{1}{2\pi h^2} e^{-\frac{||x-x_c||^2}{2h^2}}$



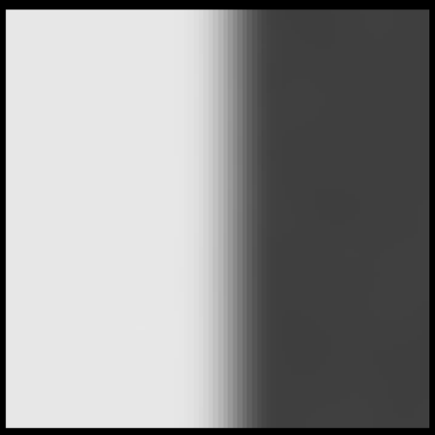
Result with a small h = 0.3

- High random error
  - i.e., high $var(\hat{f}_h(x))$
- Low systematic error
  - i.e., low $bias(\hat{f}_h(x))$

If we use a small bandwidth, say, 0.3, we fail to generate a smooth image, i.e., we have a high variance compared to its reference image containing black/white with the sharp edge.
Fortunately, we have a small bias on the edge with the small bandwidth. This means that we have a small bias error.

Bias-Variance Tradeoff

- **Image filtering**
  - Gaussian filter: $\hat{f}_h(x) = \frac{1}{2\pi h^2} e^{-\frac{||x-x_c||^2}{2h^2}}$



Result with a large h = 5

- Low random error
  - i.e., low $var(\hat{f}_h(x))$
- High systematic error
  - i.e., high $bias(\hat{f}_h(x))$

On the other hand, when we use a large bandwidth value, we get much smoother image, low variance.
Unfortunately, we get a high bias on the edge.

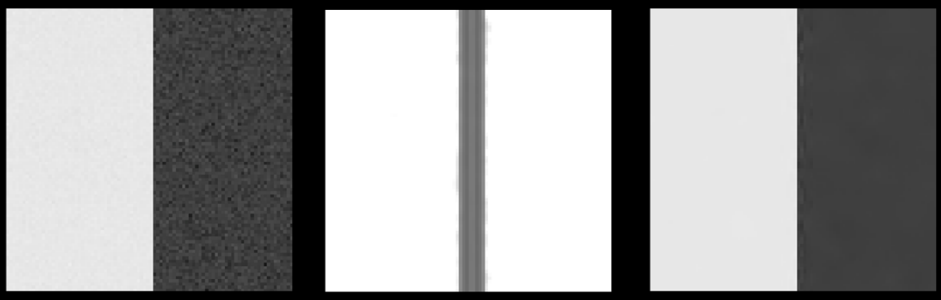
Our Approach

- **Local bandwidth selection (adaptive filtering)**
  - In each pixel, we estimate the optimal bandwidth that minimizes the following:
    - $MSE(\hat{f}_h(x)) = E[\hat{f}_h(x) - \mathrm{f(x)}]^2 + var(\hat{f}_h(x))$
    - In smooth regions,
      - Large bandwidths are desirable for reducing variance.
    - In edge regions,
      - Small bandwidths are necessary for minimizing bias.

The main problem is how to optimally choose bandwidths on different regions of the image.
Ideally, we want to set high bandwidth on smooth regions, while setting smaller bandwidth on edge regions.

Our Approach

- **Local bandwidth selection (adaptive filtering)**
  - In each pixel, we estimate the optimal bandwidth that minimizes the following:
    - $MSE\left(\hat{f}_h(x)\right) = E\left[\hat{f}_h(x) - \mathrm{f(x)}\right]^2 + var(\hat{f}_h(x))$

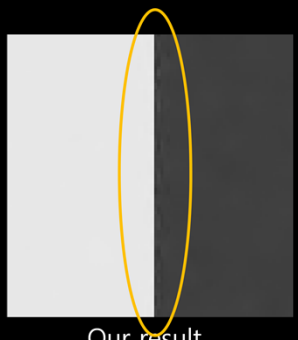


Noisy image | Our bandwidth map | Our result

Especially, we estimate the MSE for each pixel without relying upon the reference image, and compute our bandwidth map, as shown in the slide; white values in the map indicates large bandwidth values.

Our Approach

- **Local bandwidth selection (adaptive filtering)**
  - $MSE\left(\hat{f}_h(x)\right) = E\left[\hat{f}_h(x) - \mathrm{f(x)}\right]^2 + var(\hat{f}_h(x))$
- **Local sample allocation (adaptive sampling)**
  - Allocate more rays on pixels with high MSEs

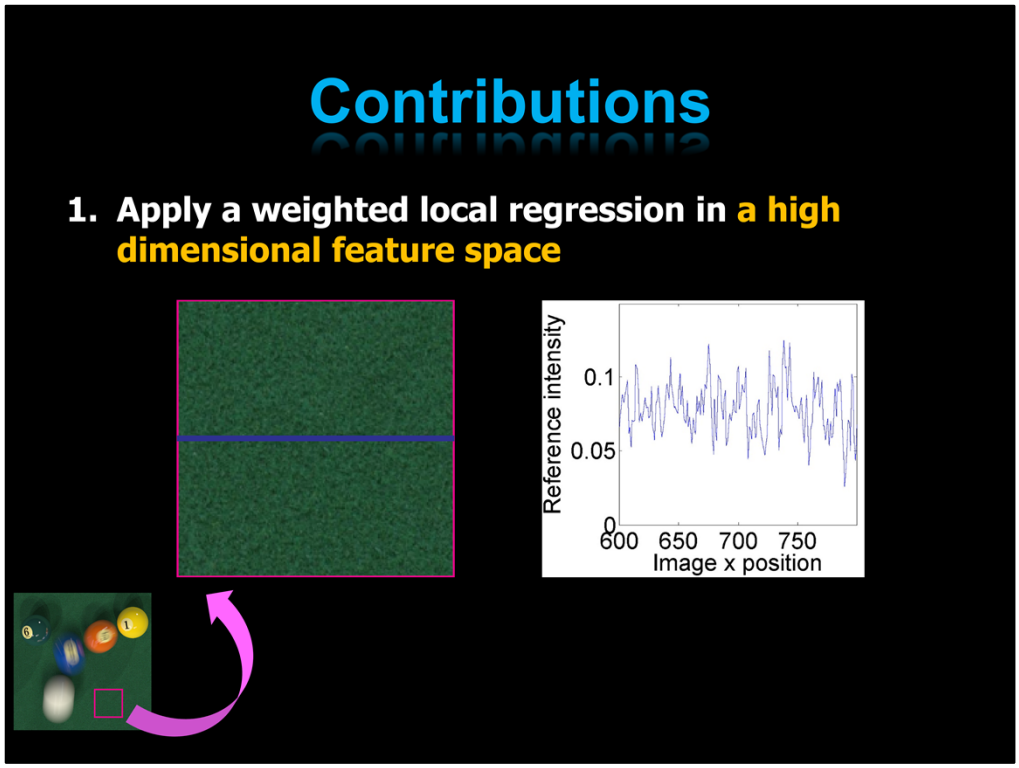


Our result

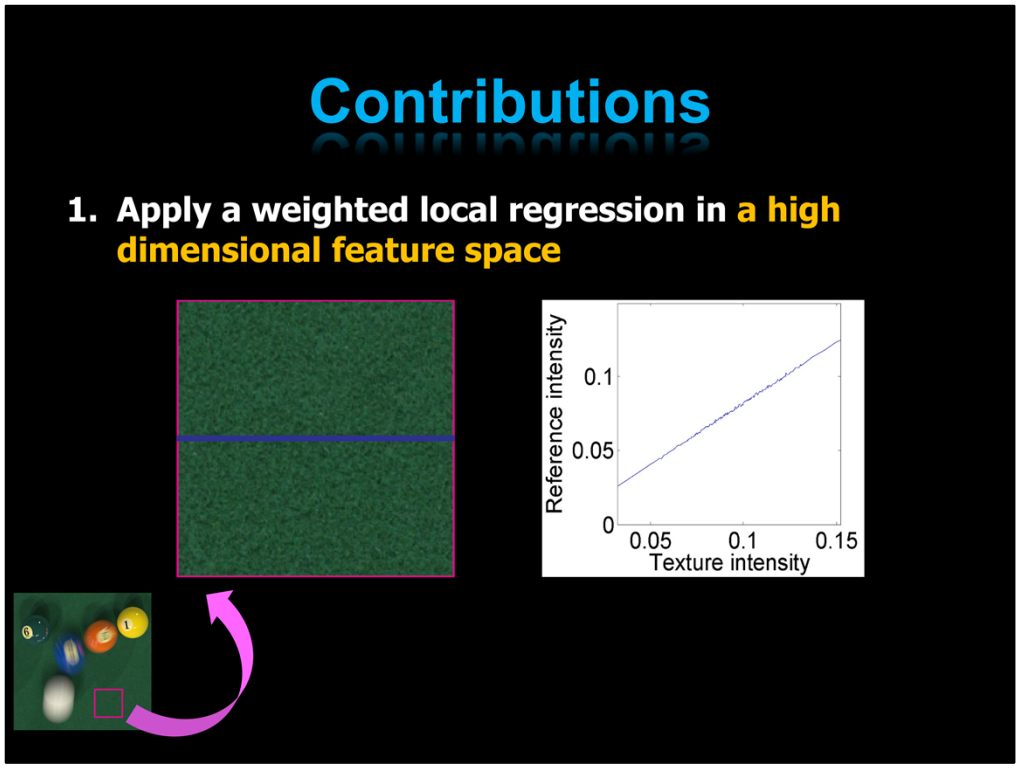
We then allocate more samples on regions with high MSE values.

Contributions

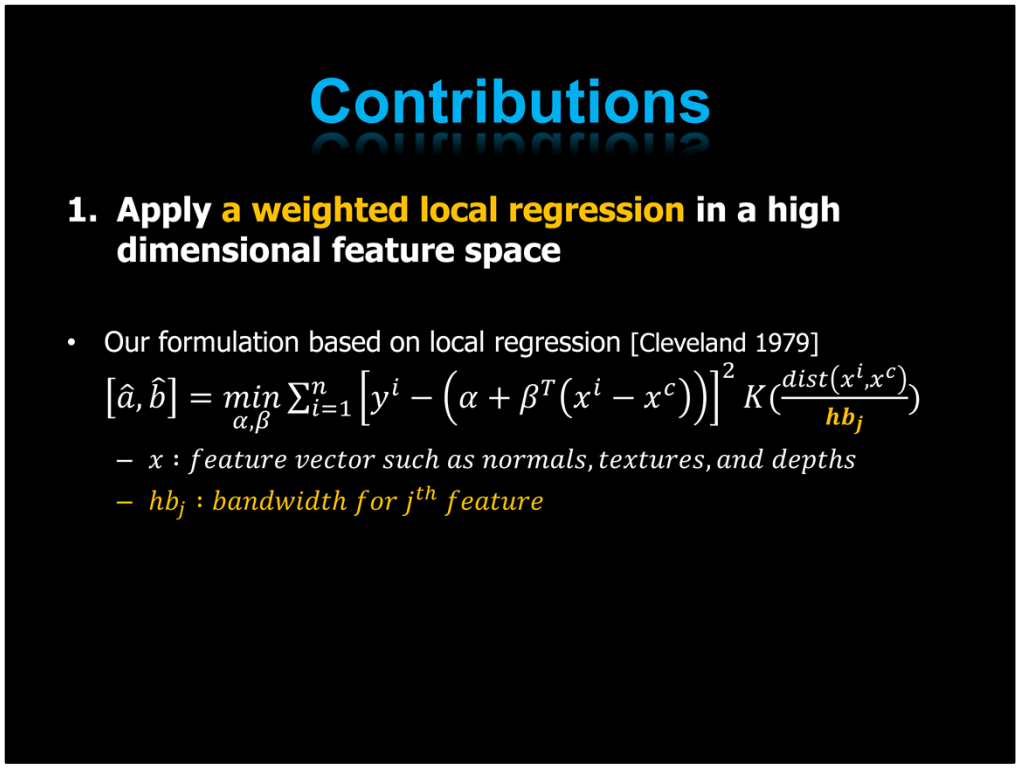
1. Apply a weighted local regression in a high dimensional feature space
2. Propose an automatic bandwidth selection to leverage different types of features



Our first observation is that image values look very noise according to image space, as a low dimensional feature space.



But, when we use a higher dimensional feature space, in this case, one including the texture intensity, the output value linearly dependent on the texture feature.



As a result, we use a high dimensional vector space, typically including image space, texture, normal, commonly available at the G-buffer.

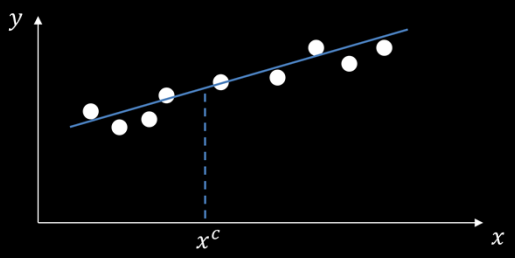
We then assume a linear regression function whose coefficient are alpha and beta.
The main question is how to optimally chose bandwidth parameters. In our formulation, we especially chose two terms, h and b_j, their products hb_j is the bandwidth for j th feature element.

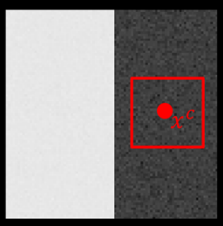
Contributions

1. **Apply a weighted local regression in a high dimensional feature space**

- Our formulation based on local regression [Cleveland 1979]

$$[\hat{a}, \hat{b}] = \min_{\alpha,\beta} \Sigma_{i=1}^{n} \left[y^i - \left(\alpha + \beta^T \left(x^i - x^c \right) \right) \right]^2 K\left(\frac{dist(x^i, x^c)}{hb_j} \right)$$





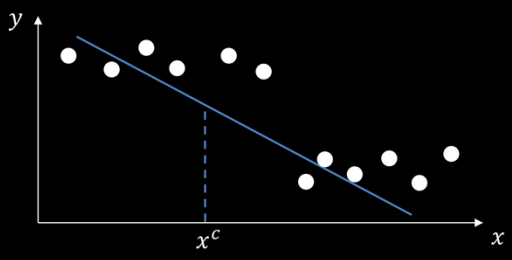
Visually speaking, we try to fit a line on MC samples.

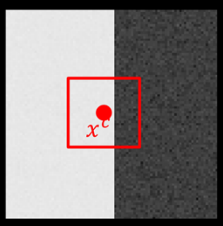
Contributions

1. **Apply a weighted local regression in a high dimensional feature space**

- Our formulation based on local regression [Cleveland 1979]

$$[\hat{a}, \hat{b}] = \min_{\alpha,\beta} \Sigma_{i=1}^{n} \left[y^i - \left(\alpha + \beta^T \left(x^i - x^c \right) \right) \right]^2 K\left(\frac{dist(x^i, x^c)}{hb_j} \right)$$



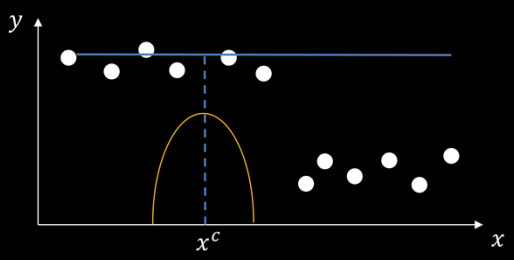


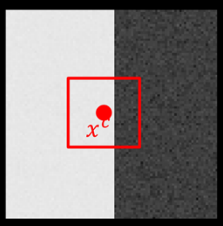
Contributions

1. **Apply a weighted local regression in a high dimensional feature space**

- Our formulation based on local regression [Cleveland 1979]

$$[\hat{a}, \hat{b}] = \min_{\alpha,\beta} \Sigma_{i=1}^{n} \left[y^i - \left(\alpha + \beta^T \left(x^i - x^c \right) \right) \right]^2 K\left(\frac{dist(x^i, x^c)}{\boldsymbol{hb_j}} \right)$$



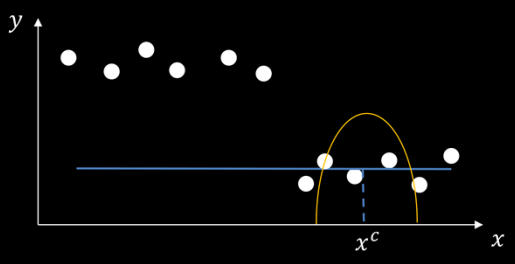


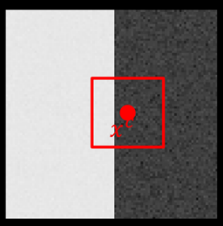
Contributions

1. **Apply a weighted local regression in a high dimensional feature space**

- Our formulation based on local regression [Cleveland 1979]

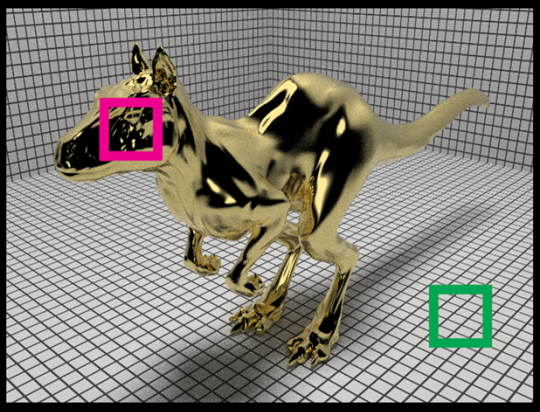
$$[\hat{a}, \hat{b}] = \min_{\alpha,\beta} \sum_{i=1}^{n} \left[y^i - \left(\alpha + \beta^T \left(x^i - x^c \right) \right) \right]^2 K\left(\frac{dist(x^i, x^c)}{hb_j} \right)$$





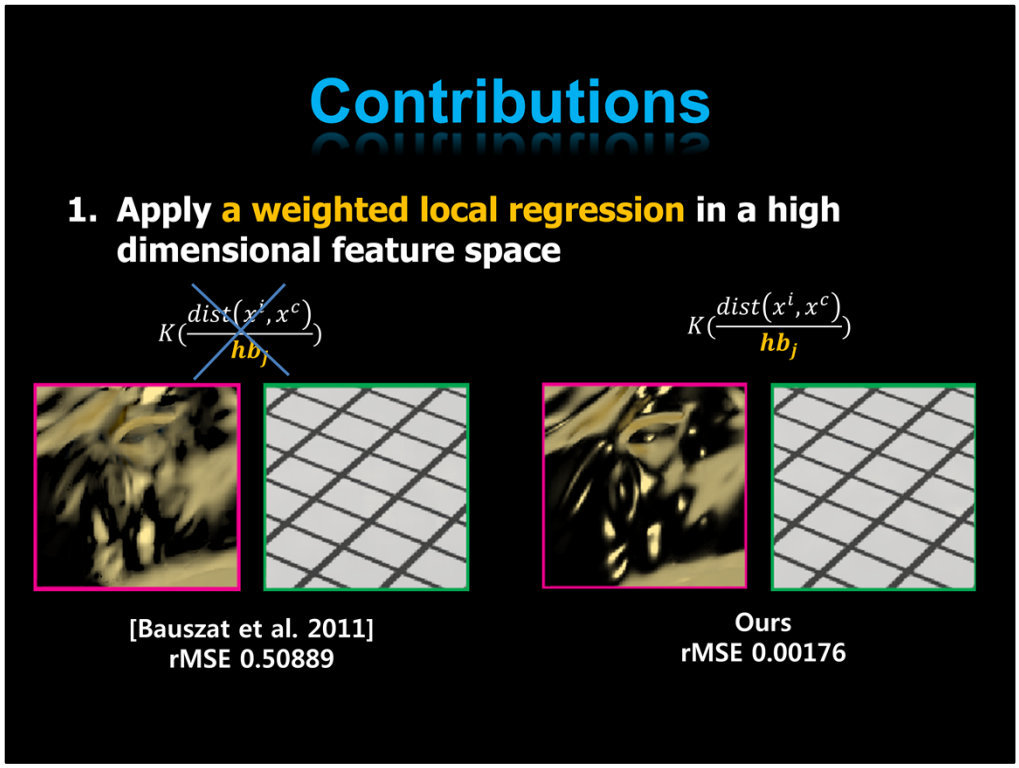
Contributions

1. Apply **a weighted local regression** in a high dimensional feature space



Input, N = 32

Killeroo model courtesy of headus 3D



These are filtered images w/ and w/o considering bandwidth parameters.

Contributions

2. **Propose an automatic bandwidth selection to leverage different types of features**

$$[\hat{a}, \hat{b}] = \min_{\alpha,\beta} \sum_{i=1}^{n} \left[y^i - \left(\alpha + \beta^T (x^i - x^c) \right) \right]^2 K\left(\frac{dist(x^i, x^c)}{\boldsymbol{hb_j}}\right)$$

- Design a new optimal bandwidth selection based on the asymptotic expression [Ruppert 1994]
  - $bias(\hat{f}_{hb}(x)) \propto 0.5h^2 trace\left(BH\hat{f}_{hb}(x)\right)$
  - $var(\hat{f}_{hb}(x)) \propto \frac{1}{n(x)h^k \prod_{j=1}^{k} b_j}$
  - Please see the thesis for the details.

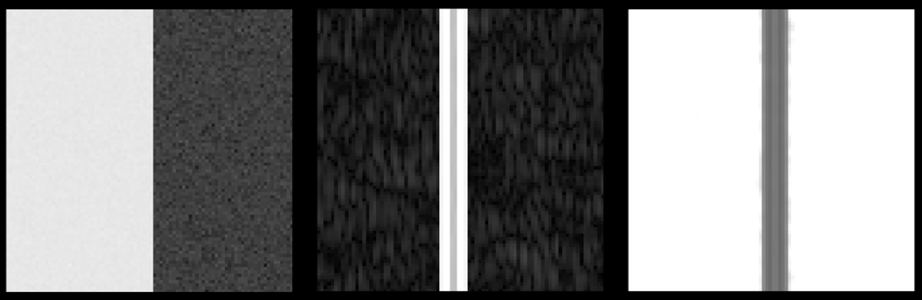
Our main theoretical contributions are on selecting bandwidth parameters minimizing the MSE values.
WE have found that bias and variance of our linear regression model has useful asymtotpic relationship.
Please refer to the paper for more details.

Contributions

2. **Propose an automatic bandwidth selection to leverage different types of features**

$$[\hat{a}, \hat{b}] = \min_{\alpha,\beta} \sum_{i=1}^{n} \left[y^i - \left(\alpha + \beta^T (x^i - x^c) \right) \right]^2 K\left(\frac{dist(x^i, x^c)}{hb_j} \right)$$

- $bias\left(\hat{f}_{hb}(x)\right) \propto 0.5h^2 trace\left(BH\hat{f}_{hb}(x)\right)$



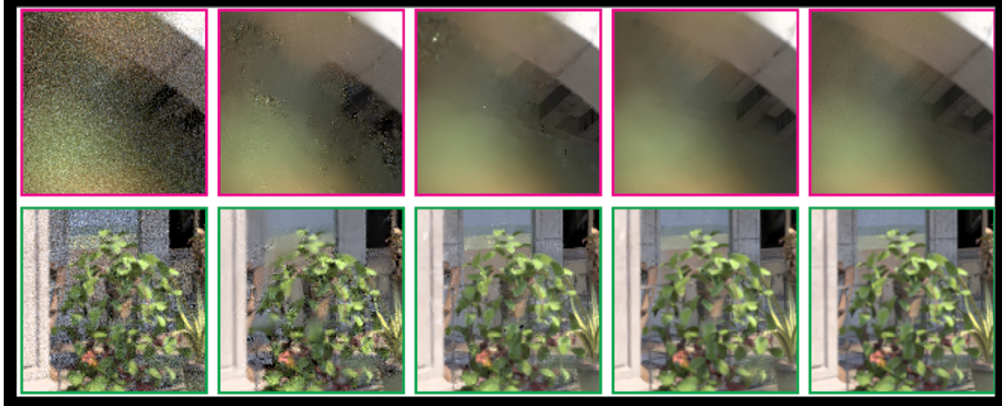
Noisy image | Estimated derivatives | Our bandwidth map

Intuitively, the bias is related to the second derivative of unknown functions. So, we set smaller bandwidth for higher second derivatives.

Results

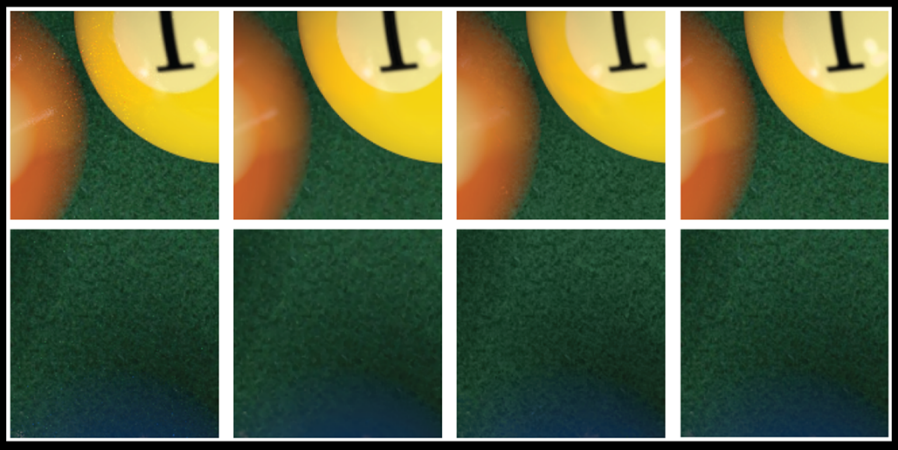
- **San Miguel**
  - Path tracing
  - Depth of field effects
  - Complex textured geometries

Equal-time Comparison



LD, N = 128 (665 s) rMSE 0.06288	NLM, N = 115 (665 s) rMSE 0.01242	SURE, N = 113 (665 s) rMSE 0.01521	Ours, N = 115 (660 s) rMSE 0.00448	Reference N = 16K

Equal-quality Comparison



NLM, N = 512 (1064 s) rMSE 0.00047	SURE, N = 1K (2202 s) rMSE 0.00063	Ours, N = 32 (91 s) rMSE 0.00044	Reference N = 64K

Equal-time Comparison for Animation

NLM [Rousselle et al. 2012]
(136 spp)

Our method
(128 spp)

Conclusion and Future Work

- **Reduce the required number of ray samples based on weighted local regression (WLR)**
- **Future work**
  - High-quality offline rendering
    - Efficient multi-dimensional adaptive rendering
  - Interactive or real-time rendering
    - Progressive image filtering
    - Robust error analysis for real-time global illumination

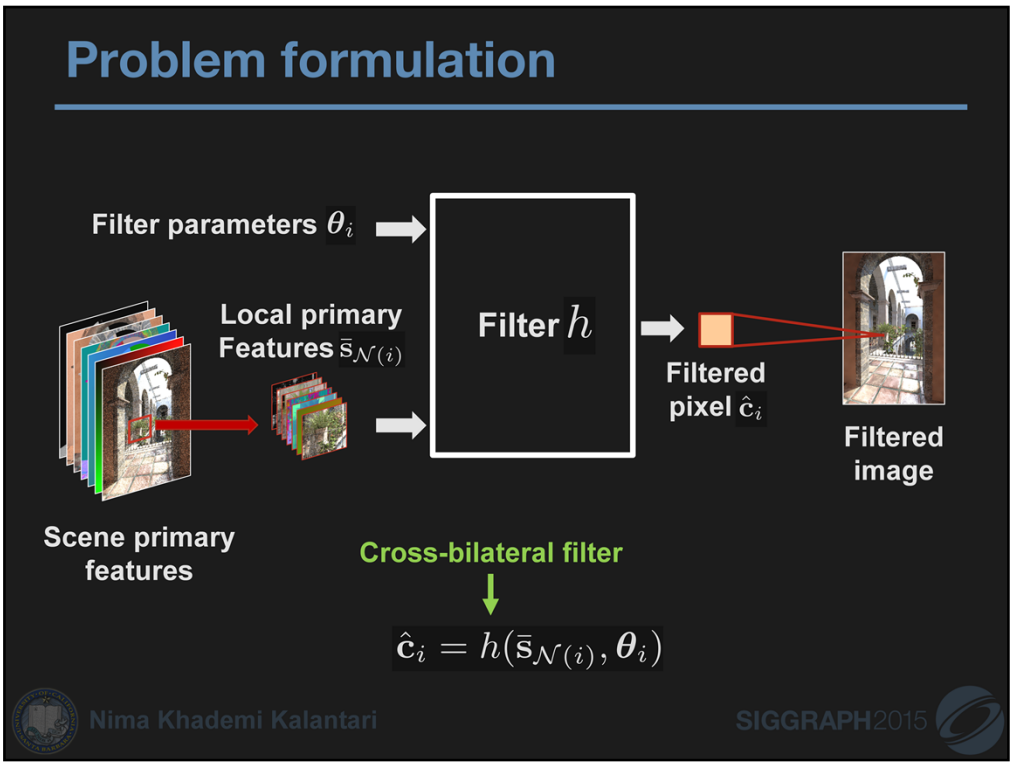
Thank you

A Machine Learning Approach for Filtering Monte Carlo Noise

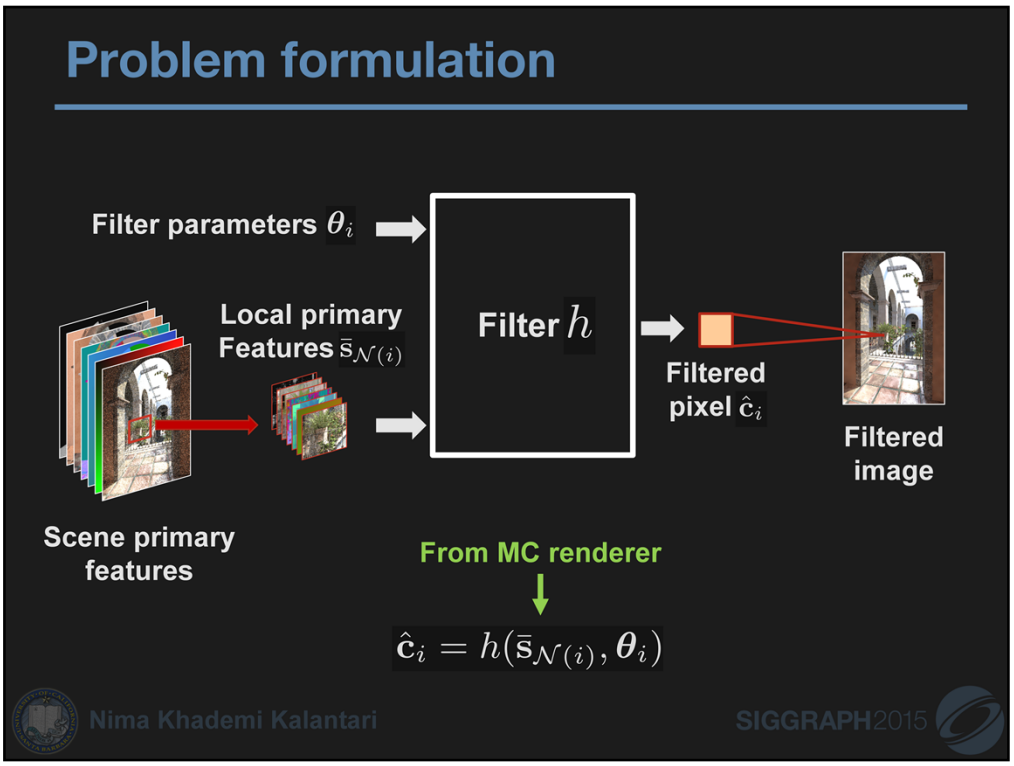
scene by Jo Ann Elliott

Nima Khademi Kalantari **Steve Bako** **Pradeep Sen**

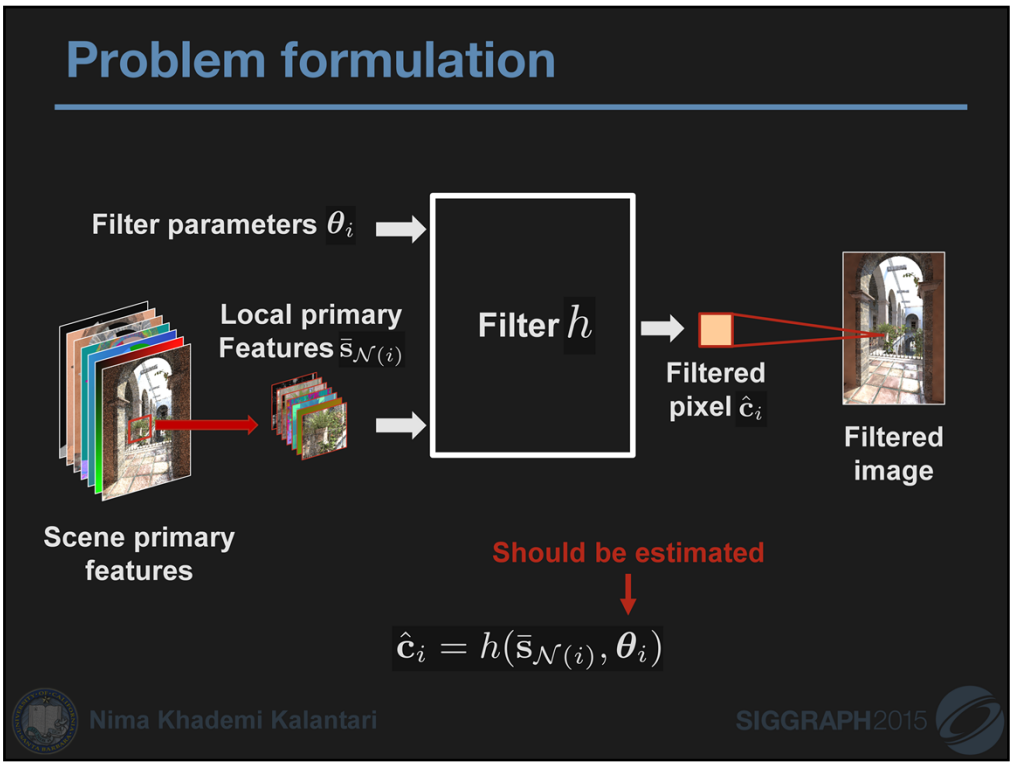
University of California, Santa Barbara



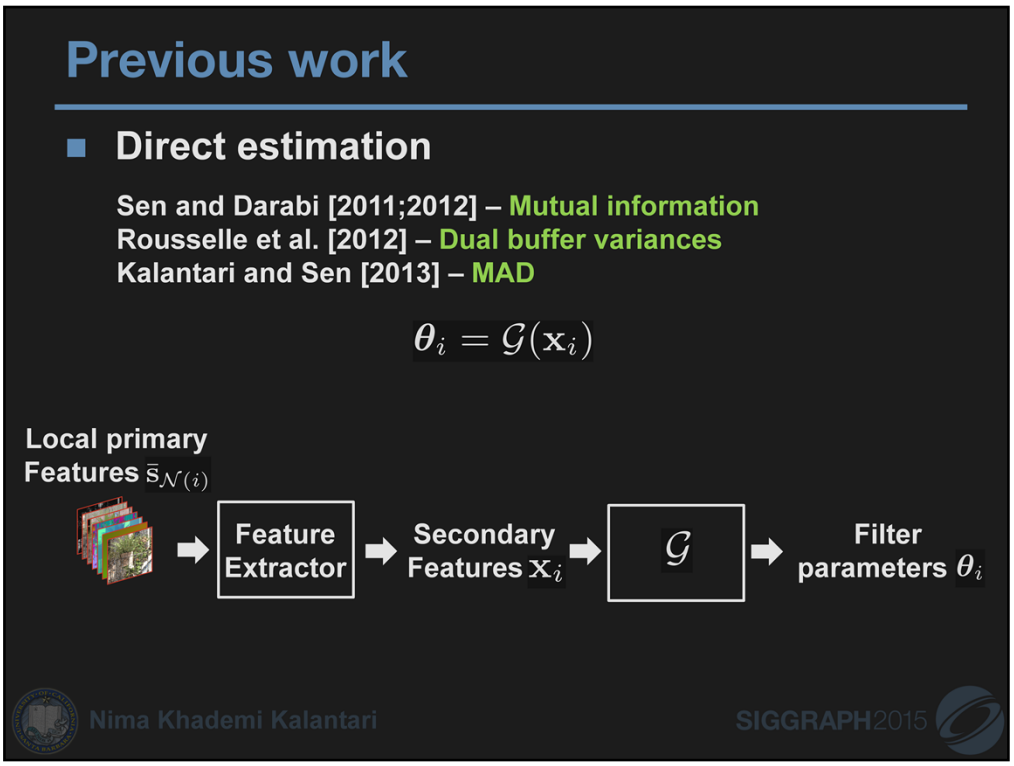
In general a filter takes local primary features as well as filter parameters to produce filtered pixel. The filter parameters are important and should be properly estimated.



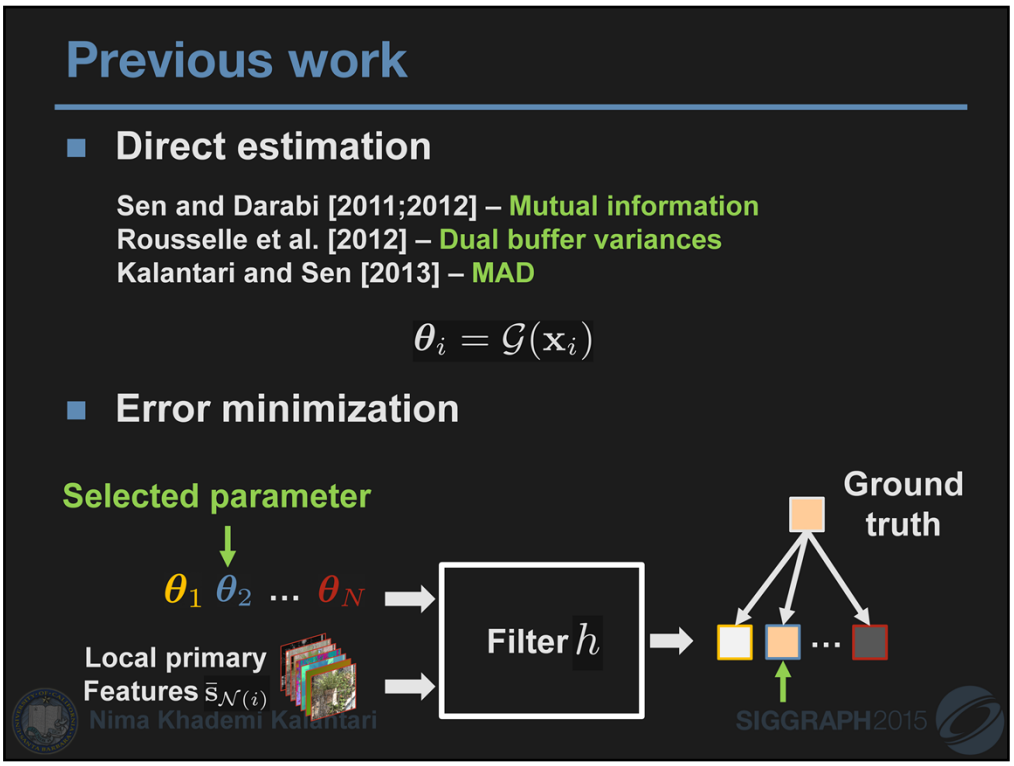
In general a filter takes local primary features as well as filter parameters to produce filtered pixel. The filter parameters are important and should be properly estimated.



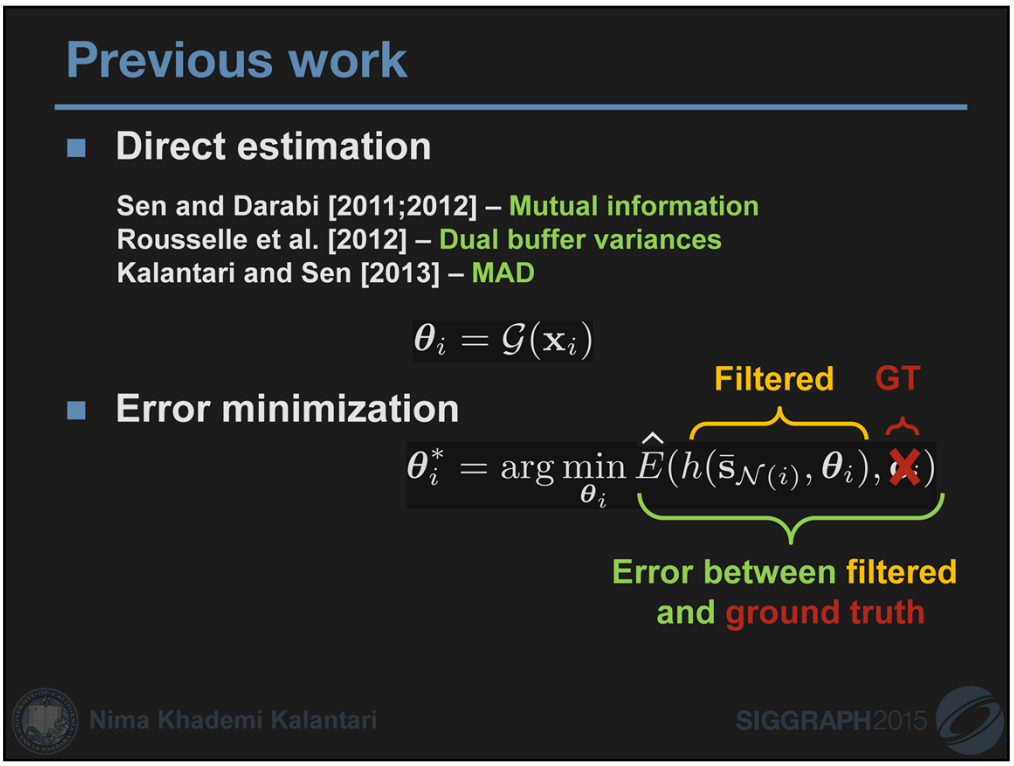
In general a filter takes local primary features as well as filter parameters to produce filtered pixel. The filter parameters are important and should be properly estimated.



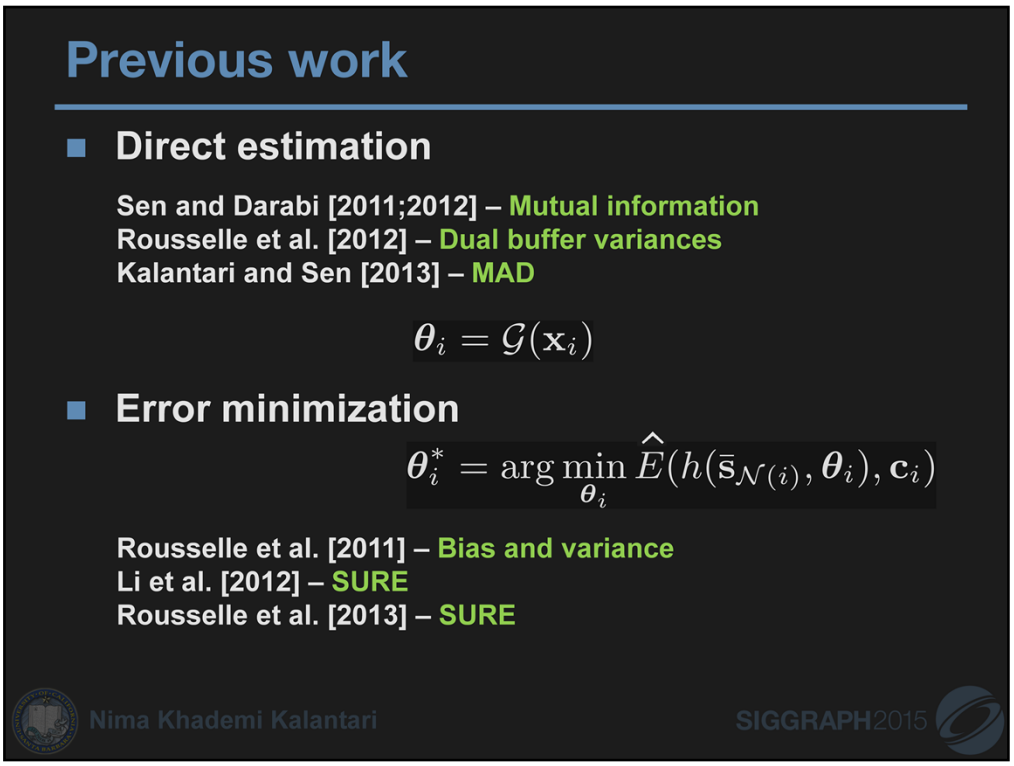
Direct estimation approaches process the local primary features to produce a set of secondary features. They then combine these secondary features through the function G to directly estimate the filter parameters.



Error minimization techniques try a set of filter parameters and choose the one with minimum error.



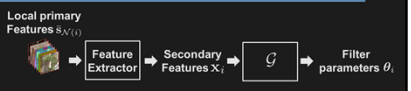
Since the ground truth pixel value is not known, they use no-reference error estimates.



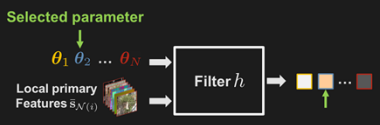
Since the ground truth pixel value is not known, they use no-reference error estimates.

Comparison

- **Direct estimation**
  - ✗ Do not minimize error
  - ✓ Estimate parameters directly
  - ✗ Define $\mathcal{G}$ heuristically
  - ✓ Do not need error metric



- **Error minimization**
  - ✓ Minimize error
  - ✗ Perform brute force search
  - ✓ Do not need $\mathcal{G}$
  - ✗ Need error metric which could be noisy

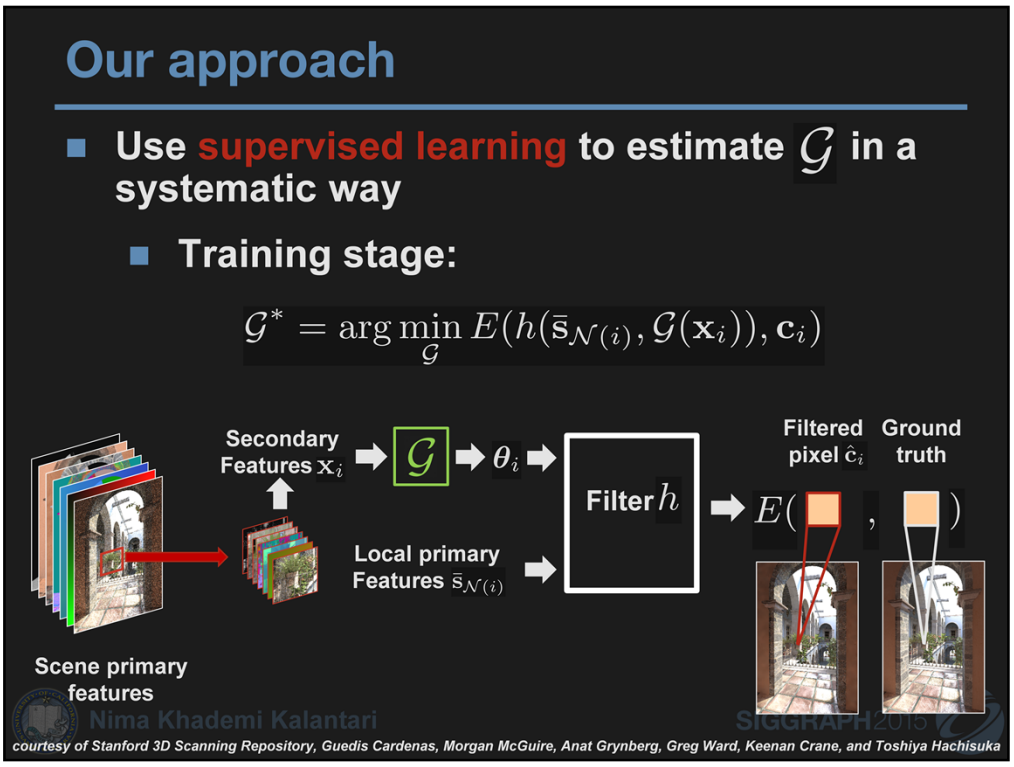


Nima Khademi Kalantari

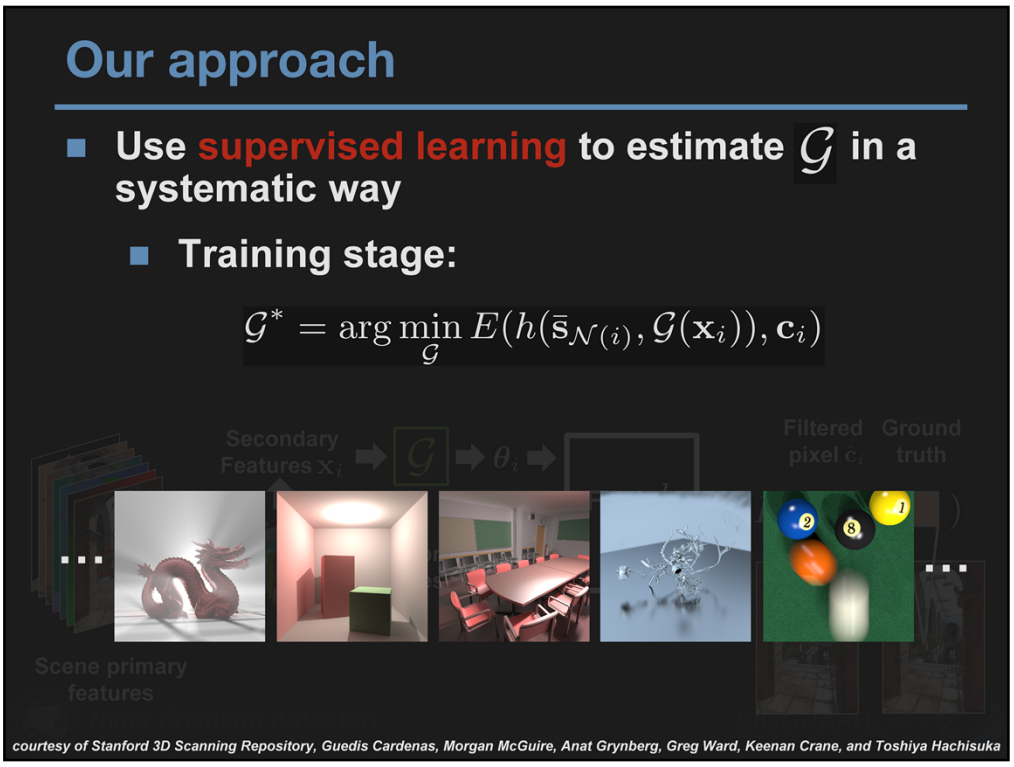
SIGGRAPH2015

Observations

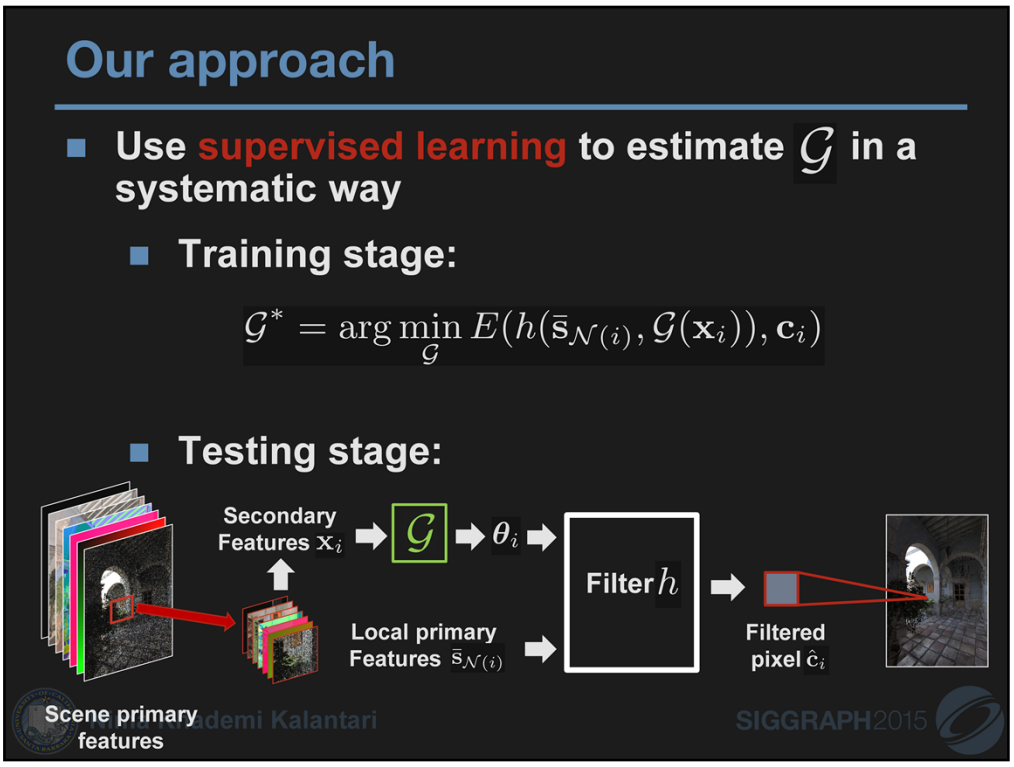
- $\mathcal{G}$ is complicated
  - Hard to define explicitly
- Filter parameters depend on many factors
  - Mutual information
  - Dual buffer variances
  - MAD



In the training stage G is estimated by minimizing the above energy function on a set of training scenes.



In the training stage G is estimated by minimizing the above energy function on a set of training scenes.

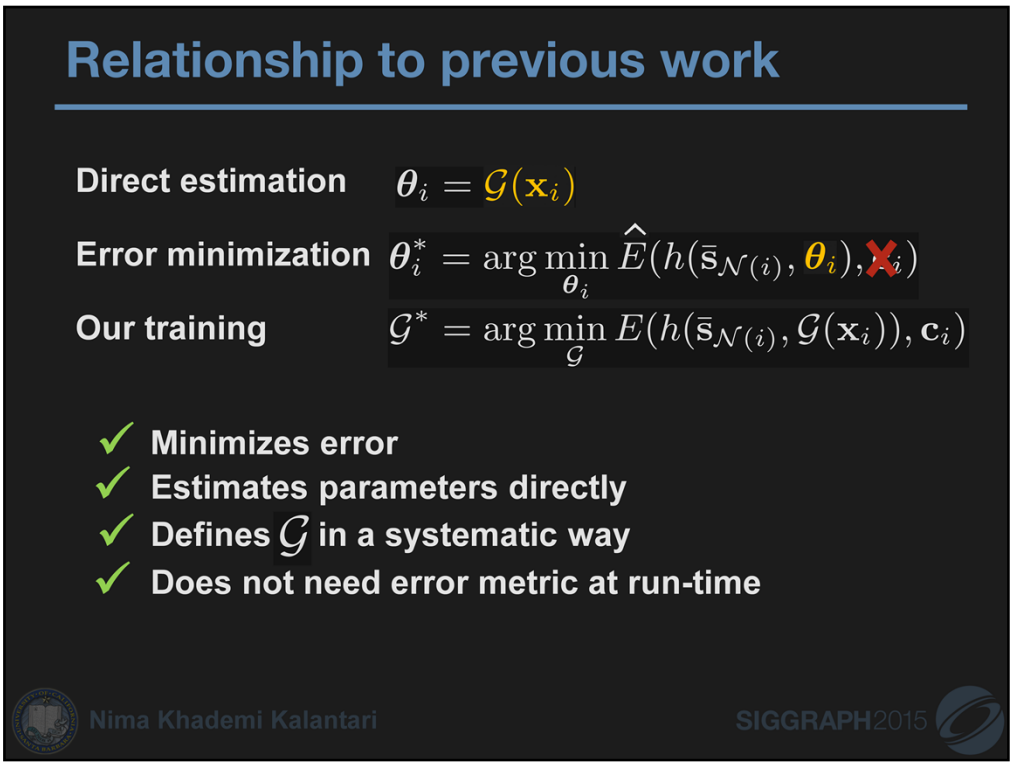


The trained G can then be used to estimate filter parameters for a new test scene and produce filtered image.

Learning system

- Representation (neural networks)
- Error metric (based on relative MSE)
- Optimization (backpropagation)

$$\mathcal{G}^* = \arg\min_{\mathcal{G}} E(h(\bar{\mathbf{s}}_{\mathcal{N}(i)}, \mathcal{G}(\mathbf{x}_i)), \mathbf{c}_i)$$



Our training energy equation has elements from both error minimization and direct estimation techniques, and thus, have all their advantages.

Results

Kitchen (GI)

4 samples/pixel

Our MLT
40.9 s

32K samples/pixel

Ground truth
~ 3 days

Scene by Jo Ann Elliott

Kitchen (GI)

4 samples/pixel

Rousselle et al. 2013 (RD)
58.9 s

32K samples/pixel

Ground truth
~ 3 days

Scene by Jo Ann Elliott

Kitchen (GI)

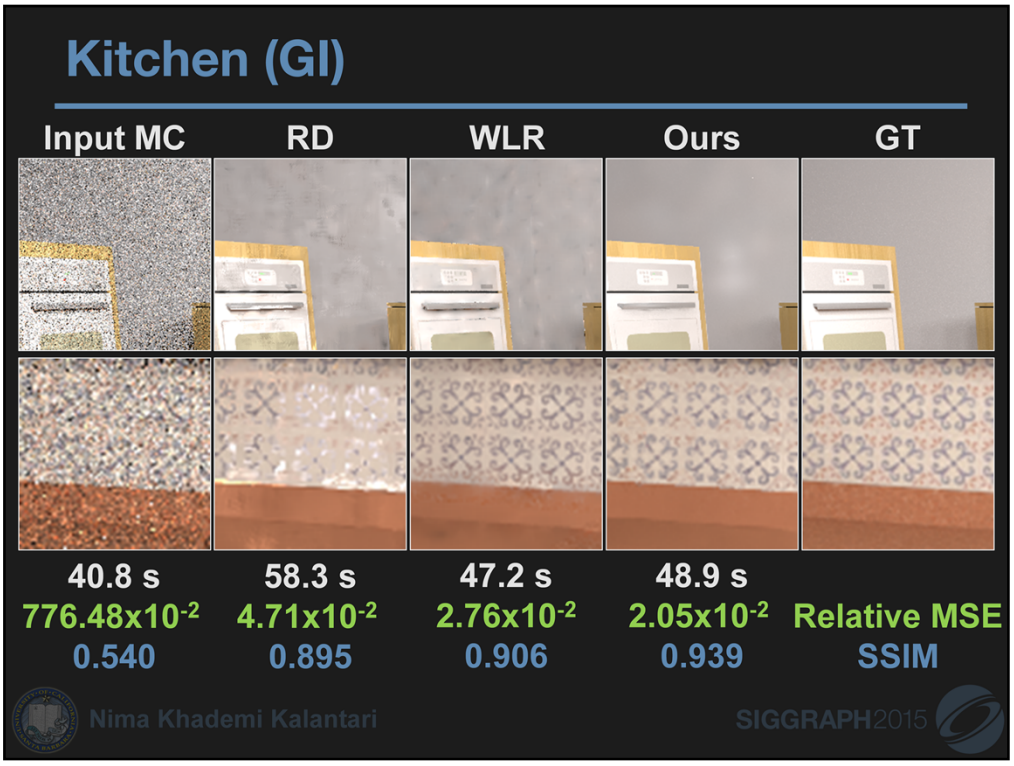
4 samples/pixel

Moon equal result (WLR)
47.2 s

32K samples/pixel

Ground truth
~ 3 days

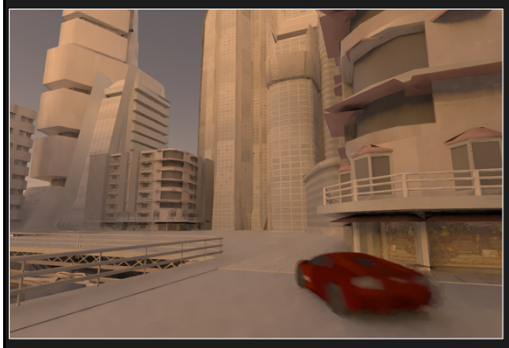
scene by Jo Ann Elliott

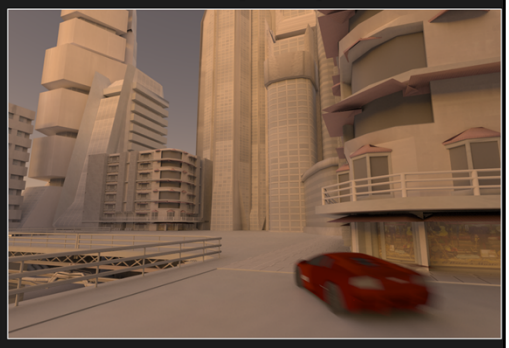

Kitchen (GI)
Input MC
RD
WLR
Ours
GT
40.8 s
776.48x10-2
0.540
58.3 s
4.71x10-2
0.895
47.2 s
2.76x10-2
0.906
48.9 s
2.05x10-2
0.939
Relative MSE
SSIM
Nima Khademi Kalantari
SIGGRAPH2015

Downtown (GI + MB)

8 samples/pixel

4K samples/pixel





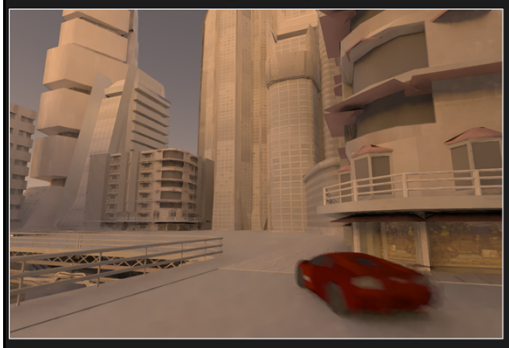
Ground truth
~ 6 hours

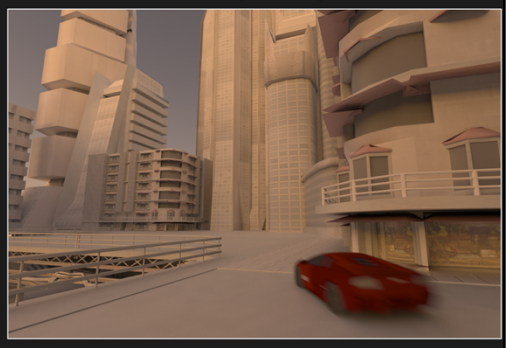
scene by Herminio Nieves and tf3dm.com user ysup12

Downtown (GI + MB)

8 samples/pixel

4K samples/pixel





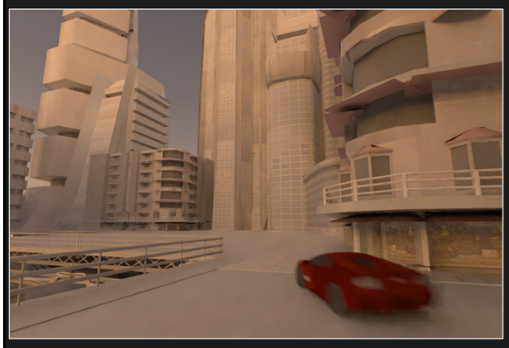
Rousselle et al. 2013 (RD)
74.9 s

Ground truth
~ 6 hours

scene by Herminio Nieves and tf3dm.com user ysup12

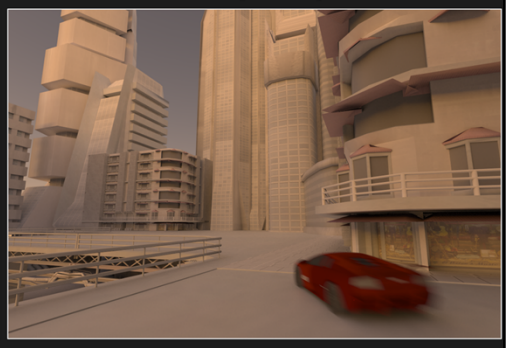
Downtown (GI + MB)

8 samples/pixel



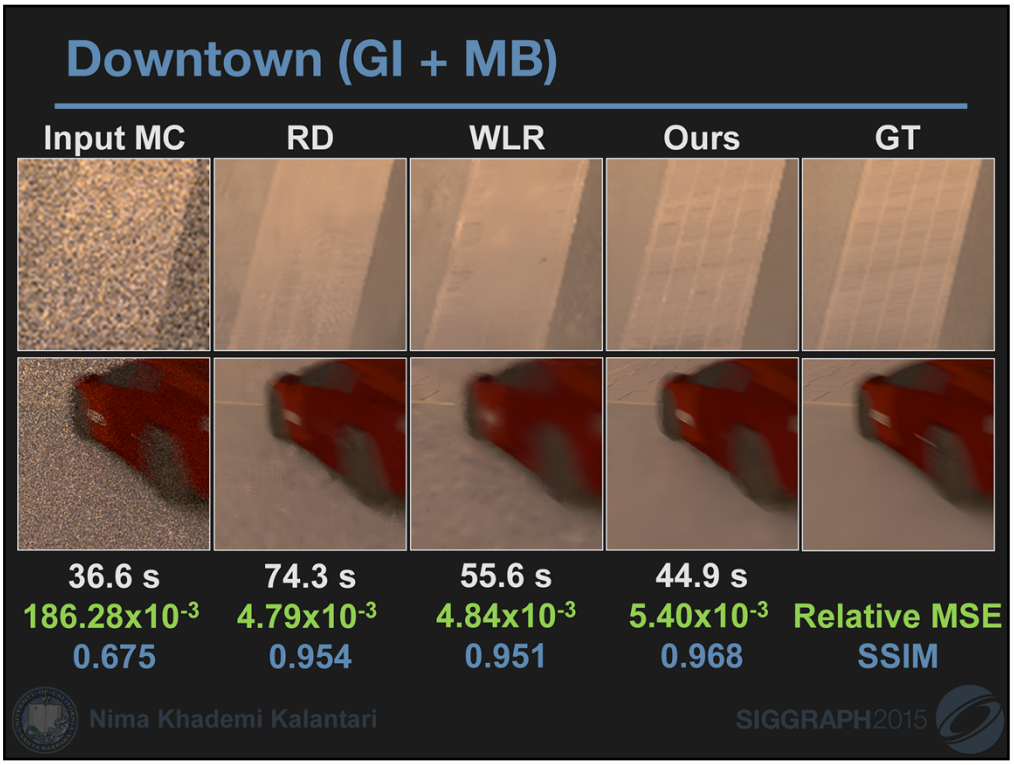
Moon equation (WLR)
55.0 s

4K samples/pixel



Ground truth
~ 6 hours

scene by Herminio Nieves and tf3dm.com user ysup12


Downtown (GI + MB)
Input MC
RD
WLR
Ours
GT
36.6 s
186.28x10-3
0.675
74.3 s
4.79x10-3
0.954
55.6 s
4.84x10-3
0.951
44.9 s
5.40x10-3
0.968
Relative MSE
SSIM
Nima Khademi Kalantari
SIGGRAPH2015

Animated sequences


Kitchen
Nima Khademi Kalantari
SIGGRAPH2015

Input MC 8 spp (80 sec/frame)

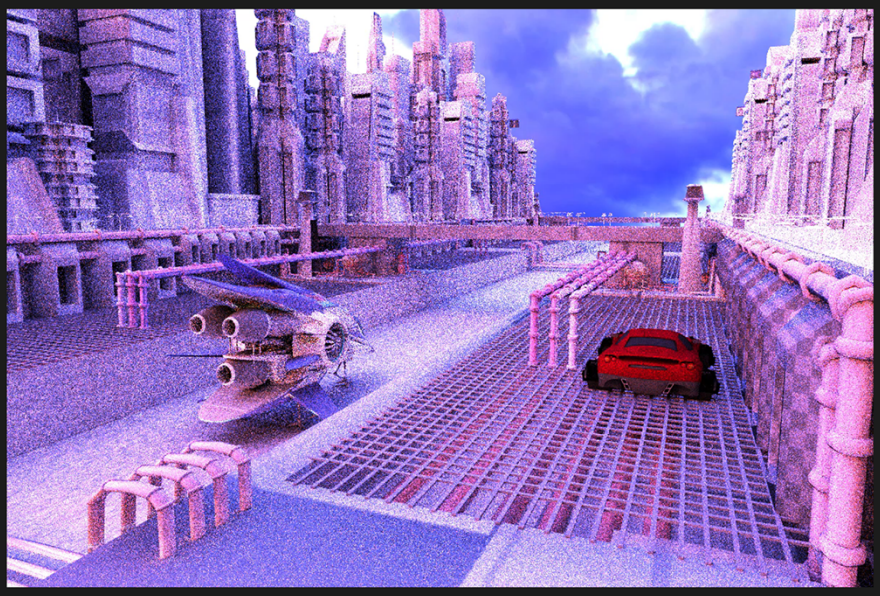
scene by Jo Ann Elliott

Our result 8 spp (145 sec/frame)

scene by Jo Ann Elliott

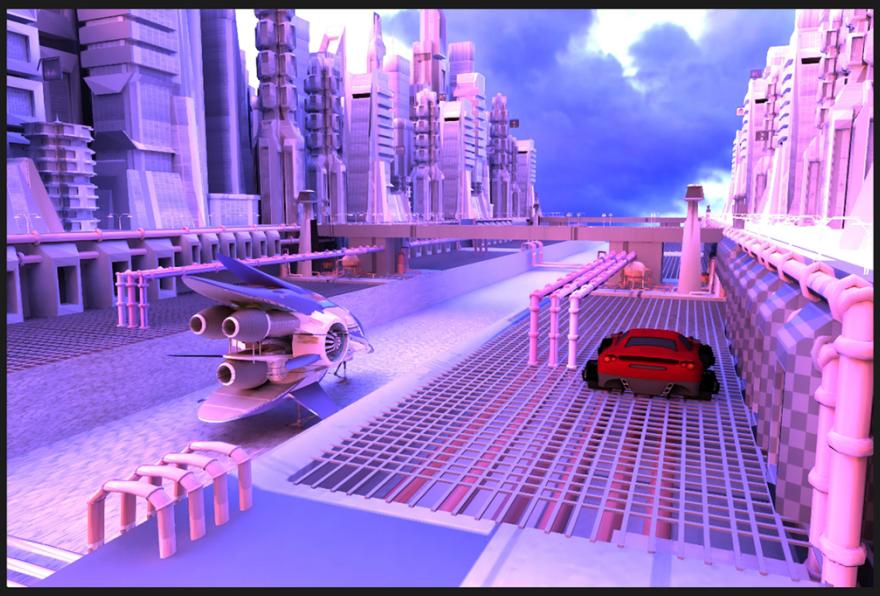

Sci-fi city
Nima Khademi Kalantari
SIGGRAPH2015

Input MC 16 spp (70 sec/frame)



scene by Herminio Nieves and tf3dm.com user ysup12

Our result 16 spp (135 sec/frame)



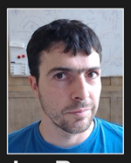
scene by Herminio Nieves and tf3dm.com user ysup12

Denoising Your Monte Carlo Renders:

Recent Advances in Image-Space Adaptive Sampling and Reconstruction

Pradeep Sen
UC Santa Barbara

Matthias Zwicker
University of Bern



Fabrice Rousselle
Disney Research

Sung-Eui Yoon
KAIST

Nima Khademi Kalantari
UC Santa Barbara

SIGGRAPH2015

CONCLUSIONS

scene by Jesper Lloyd

SIGGRAPH2015

Conclusions

- We have just surveyed a set of recent algorithms for MC denoising
- Tip of the iceberg – an active area of research
- STAR paper [Zwicker et al. 2015] documents great progress over last 5-10 years

Adoption by industry

- Films using Monte Carlo denoising for production:
  - *Elysium*, TriStar Pictures 2013
  - *Big Hero 6*, Disney 2014
- Innobright – new startup developing MC denoising plugins for different renderers
  - P. Sen and M. Zwicker in technical advisory board

Open challenges

- Real-time applications
- Animation sequences
- Integration of *a priori* and *a posteriori* techniques
- Leverage theoretical foundations
  - Learning-based techniques
  - Sparse methods
- Theoretical analysis, proof of lower bounds on sampling density

Get involved!

- Source code available for all methods discussed
- Full list available in course notes and STAR paper

Acknowledgments

- Funding
  - P. Sen: NSF IIS 08-45396, 13-42931
  - M. Zwicker: SNSF 143886
  - S.E. Yoon: NRF 2013R1A1A2058052
- Co-authors on all papers
- Monte Carlo rendering community